# FINAL REPORT

## High-Accuracy Multisensor Geolocation Technology to Support Geophysical Data Collection at MEC Sites

## SERDP Project MR-1564

DECEMBER 2012

Dorota Grejner-Brzezinska
Charles Toth
**The Ohio State University**

Andrey Soloviev
**University of Florida**



SERDP
DOD ▪ EPA ▪ DOE

# Report Documentation Page

*Form Approved*
*OMB No. 0704-0188*



| 1. REPORT DATE | 2. REPORT TYPE | 3. DATES COVERED |
|---|---|---|
| **DEC 2012** | | **00-00-2012 to 00-00-2012** |

| 4. TITLE AND SUBTITLE | 5a. CONTRACT NUMBER |
|---|---|
| **High-Accuracy Multisensor Geolocation Technology to Support Geophysical Data Collection at MEC Sites** | 5b. GRANT NUMBER |
| | 5c. PROGRAM ELEMENT NUMBER |
| 6. AUTHOR(S) | 5d. PROJECT NUMBER |
| | 5e. TASK NUMBER |
| | 5f. WORK UNIT NUMBER |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **The Ohio State University,154 W 12th Avenue,Columbus,OH,43210** | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | 10. SPONSOR/MONITOR'S ACRONYM(S) |
| | 11. SPONSOR/MONITOR'S REPORT NUMBER(S) |

12. DISTRIBUTION/AVAILABILITY STATEMENT
**Approved for public release; distribution unlimited**

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT **unclassified** | b. ABSTRACT **unclassified** | c. THIS PAGE **unclassified** | **Same as Report (SAR)** | **74** | |

# Contents

1. INTRODUCTION ..... 5
2. OVERVIEW OF THE DEEPLY INTEGRATED ARCHITECTURE ..... 8
2.1 The Concept ..... 8
2.2 The Measurement Model ..... 12
3. TEST AREA ..... 19
4. PERFORMANCE ANALYSIS ..... 23
4.1 Columbus Data Analysis ..... 23
4.2 Wayne National Forest Data Analysis ..... 25
5. GEOLOCATION NAVIGATION SOFTWARE MODULES ..... 37
5.1 AIMS PRO QUADRUPLE INTEGRATION NAVIGATION SOFTWARE ..... 37
5.1.1 Introduction ..... 37
5.1.2 System Settings ..... 39
5.1.3 Sensor Fusion ..... 42
5.2 DEEP INTEGRATION NAVIGATION SOFTWARE ..... 44
5.2.1 Introduction ..... 44
5.2.2 Sensor Fusion ..... 45
6. LASER SENSOR DEVELOPMENTS ..... 48
6.1 Introduction ..... 48
6.2 Kinect Sensor ..... 49
6.3 Sensor Repeatability Test ..... 51
6.4 Plane Fitting Performance ..... 52
6.5 Sphere Fitting Performance ..... 54
7. CONCLUSION FUTURE RECOMMENDATION ..... 60
8. REFERENCES ..... 61
9. APPENDIX ..... 63
9.1 AIMS PRO: Major functions ..... 63
9.2 AIMS PRO: Example ..... 65
9.3 Software Modules Developed (digital version) ..... 73
9.3.1 AIMS PRO Software ..... 73

9.3.2 Deep Integration Software..... 73

## EXECUTIVE SUMMARY

To improve the algorithmic robustness and to reduce the deployable hardware size and weight of the precise geolocation technology developed under the project MM-1564, which is based on quadruple sensor integration, a deeply integrated GPS signal processing solution has been developed and implemented.

In Phase I of project MM-1564, quadruple sensor-base integrated system architecture has been designed and implemented, according to the original SOW (Statement Of Work). This unique architecture combines measurements of the Global Positioning System (GPS), inertial navigation system (INS), pseudolites (PLs), and a terrestrial laser scanner (TLS). State-of-the-art GPS navigation allows for cm-accurate positioning in open areas where a sufficient number of unobstructed GPS signals is available. However, in covered areas (such as under dense tree foliage) a significant GPS signal attenuation may occur. As a result, commercial of-the-shelf GPS receivers fail to provide reliable signal measurements. Therefore, the GPS navigation technology is augmented in our solution by high-grade inertial navigation, PLs, and TLS.

In Phase II of project MM-1564, deeply integrated GPS signal processing was incorporated into the quadruple navigation solution in order to enhance the navigation robustness and, optionally, to relax the system operational requirements including the complexity of its installation and maintenance, as well as the overall cost, size, weight and power consumption. Deeply integrated GPS signal processing is an emerging technology that has been actively developed over the past three-four years. The deep integration technology maintains a complete GPS signal tracking status (code phase, carrier frequency, carrier phase and navigation data recovery) in challenging signal environments, such as urban canyons and under dense foliage. This technology exploits very long coherent integration of GPS signals; coherent integration intervals of one second, as opposed to 20 ms signal integration conventionally utilized by GPS receivers. Long coherent integration is achieved by using inertial data for dynamic aiding of the GPS signal integration function and by applying an energy-based search algorithm for the removal (wipe-off) of navigation data bits. Deep integration had been demonstrated to enable processing of GPS signals in urban canyons, indoor environments, dense forestry areas, and in the presence of wide-band sources of radio-frequency (RF) interference. The unique feature of the deep integration approach is its ability to consistently track carrier phase of GPS signals that are attenuated by 30 dB from their open sky conditions. Maintaining carrier phase tracking for weak GPS signals is crucial for enabling cm-accurate geolocation capabilities in challenging GPS environments.

To effectively support detection and discrimination of unexploded ordnance, the target accuracy goal for the geolocation technology was set to 10 cm (95%) for the absolute accuracy and 1 cm (95 %) for the relative accuracy, which performance level has been reached, as it was demonstrated in several tests. This report demonstrates the performance of a deeply integrated GPS/inertial architecture under extreme canopy coverage, where traditional receiver technology fails to acquire and track GPS signals. GPS and inertial experimental data collected in various canopy coverage areas are applied to evaluate the system performance. Test results demonstrate that the deeply integrated approach enables reliable trajectory reconstruction capabilities and maintains sub-meter level of differential positioning precision for cases where the GPS signal is severely attenuated by the canopy.

# 1. INTRODUCTION

The overall goal of the research and development effort documented in this report is to develop precise (i.e., centimeter to sub-decimeter) geolocation technology that can support detection and discrimination of unexploded ordnance. The detection and remediation of munitions and explosives-of-concern (MEC) on ranges, munitions burning and open detonation areas, and burial pits is one of the US Department of Defense's (DoD) most pressing environmental problems. The MEC characterization and remediation activities using currently available technologies often yield unsatisfactory results, and are extremely expensive, due mainly to the inability of current technology to detect all MEC present at a site, and the inability to discriminate between MEC and nonhazardous items that is primarily due to insufficient precision of georeferencing of the geophysical images. To address the precise georeferencing aspect, quadruple multi-sensor fusion architecture has been developed previously (Grejner-Brzezinska et. al., 2011). This architecture combines measurements of Global Positioning System (GPS), inertial navigation system (INS), pseudolites (PLs), and a terrestrial laser scanner (TLS). Kinematic GPS navigation solution allows for cm-precise positioning in open areas where a sufficient number of unobstructed GPS signals is available. However, in covered areas, such as dense tree foliage, a significant GPS signal attenuation is present. As a result, commercial of-the-shelf GPS receivers fail to provide reliable signal measurements. Therefore, the GPS navigation technology is augmented by high-grade inertial navigation, PLs, and TLS.

The effort reported herein extends the previously developed approach by investigating into the use of deeply integrated GPS/inertial technology in dense forestry areas. The idea is to incorporate deeply integrated GPS signal processing into the quadruple navigation solution in order to enhance the navigation robustness and relax the system operational requirements, including the complexity of its installation and maintenance, as well as the overall cost, size, weight and power consumption. Deep integration enables GPS measurements under severe signal attenuation, such as a 30 dB or a factor of 1000 power attenuation from open sky conditions. Augmenting the system with the capability to maintain GPS measurements for very weak signals increases significantly the GPS coverage in difficult environments, such as dense canopy areas. Increased availability of GPS measurements improves the robustness of the navigation solution. Specifically, additional GPS measurements can be efficiently exploited for

the identification of GPS and PL signals that are corrupted by multipath reflections from surrounding trees. Increased GPS availability will also enable reduction of the system operational requirements. Particularly, it is expected that the incorporation of deep integration will decrease the use of pseudolite-based navigation, up to a complete elimination of the pseudolites, in order to reduce the cost. In addition, deep integration should enable reduction in deployment complexity of the TLS component, and, overall, reduce the requirements of inertial sensors' performance.

Two key technological enablers of the deeply integrated GPS/inertial architecture include:

a) open-loop software architecture for robust signal tracking in challenging GPS environments (van Graas et al., 2009)

b) deep integration of GPS signal tracking with inertial navigation (Soloviev et al., 2008) for recovery of GPS signals in extreme attenuation conditions

To enable robust tracking of GPS signals in challenging environments and under changing conditions, open-loop software receiver architecture is applied as opposed to traditional closed-loop solutions. The receiver operates with batches of GPS signal samples. For each batch, the receiver estimates GPS signal parameters that include code phase, carrier frequency, and carrier phase. The open-loop tracking architecture first acquires the GPS signal using a fast Fourier transform (FFT)-based parallel search and then fine zooms on signal parameters. The open loop approach is particularly beneficial for robust tracking in challenging signal environments. The FFT-based correlation component reconstructs the accumulated complex GPS amplitude as a function of amplitude vs. code phase and Doppler shift from the FFT search space and then estimates signal parameters, including code phase, Doppler frequency shift and carrier phase, directly from this function. As a result, the signal can be observed immediately after it reappears after a temporary loss and its tracking status can be immediately recovered.

To enable processing of attenuated GPS signals, the open-loop signal approach is augmented by long coherent integration. Coherent signal integration is performed by fusing GPS signal processing with measurements of inertial sensors. Deep integration has been demonstrated to maintain a complete tracking status (i.e., tracking of the code phase, carrier frequency, carrier

phase and recover of navigation data bits) of weak GPS signals with the carrier-to-noise ratio (C/No) as low as 15 dB-Hz (Soloviev et al., 2008).

To evaluate performance of the deeply integrated solution in forestry areas, test data were collected in suburban tree-covered area of Columbus, Ohio, as well as in dense canopy environments of Wayne National Forest, Athens County, Ohio.

The remainder of the document is organized as follows. First, architecture of the deeply integrated solution is reviewed. Next, test setup is described. Finally, test performance results are presented. In addition, feasibility tests were carried out to assess the applicability of Flash LiDAR technology, as an implementation device for the TLS component of the prototype geolocation system.

## 2. OVERVIEW OF THE DEEPLY INTEGRATED ARCHITECTURE

Deep integration provides a self-contained high-sensitivity tracking solution that maintains a complete tracking status for GPS signals with very low C/No values[1]. Particularly, CA-code phase, GPS Link 1 (L1) frequency and carrier phase tracking is maintained for a 15 dB-Hz C/No level or, equivalently, for the received signal power of -160 dBm. The 15 dB-Hz tracking threshold corresponds to the tracking threshold of state-of-the-art high sensitivity COTS receivers. However, contrary to the deep integration technology, state-of-the-art COTS GPS receivers do not support a complete tracking status in the high-sensitivity mode (generally, only the CA-code tracking is maintained, while the carrier tracking function is not provided) and often require external aiding for high-sensitivity GPS operations (e.g., aiding of navigation data bits).

### 2.1 The Concept

Figure 2.1 shows a high-level diagram of the deeply integrated GPS receiver technology (Soloviev et al., 2008).

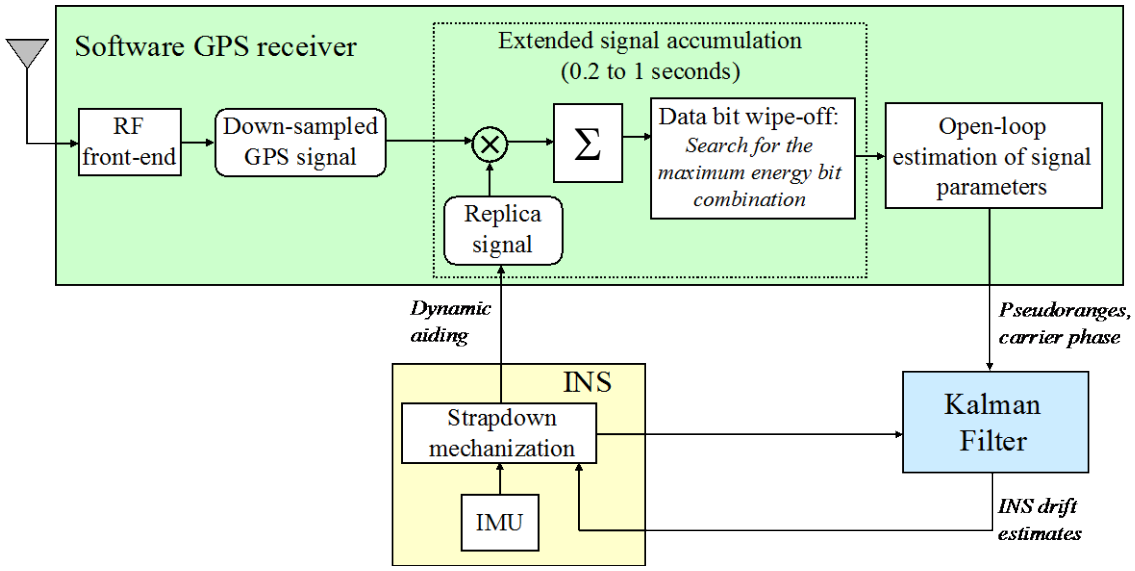



**Figure 2.1** Generic architecture of the deeply integrated GPS/inertial solution

The deep integration approach eliminates conventional tracking loops and starts with the fusion of GPS and inertial data at the earliest processing stage possible by combining RF GPS samples

[1] It is computed as $10*\log_{10}(E*B/sigma\_noise^2)$, where E is the signal energy and B is the bandwidth.

with sampled inertial measurements. Inertial data provide the dynamic reference trajectory for the GPS signal integration inside GPS receiver correlators. Particularly, parameters of the internally generated replica GPS signal are adjusted for dynamic changes using the inertial aiding. Correlator outputs are used to estimate GPS signal parameters that include CA-code phase shift, carrier Doppler frequency shift, and carrier phase. Estimated GPS signal parameters are applied to periodically calibrate the INS error states to maintain a sufficient accuracy of the inertial aiding of GPS signal integration.

Coherent signal integration over 1 s intervals is applied to recover very weak GPS signals (at a 15 dB-Hz level). The coherent integration time of 1 s exceeds the duration of data bits (0.02 s) in the GPS navigation message. A bit wipe-off is thus required to avoid energy losses during signal accumulation. A computationally efficient algorithm was, therefore, developed to search through possible bit combinations and choose the combination that maximizes the signal energy (Soloviev et al., 2009). Consequently, no external bit aiding is required.

The open-loop tracking approach computes GPS signal measurements from signal accumulation results. During the signal accumulation, a 3D GPS signal function (which is a complex signal amplitude vs. possible values of code shift and Doppler frequency shift) is constructed for every accumulation epoch. The signal parameters are then estimated directly from this function without employing traditional tracking loops. The main advantage of the open-loop tracking, as compared to traditional closed-loop techniques, is in the optimized robustness of GPS signal processing, which is especially beneficial in adverse environments such as urban (indoor and outdoor), dense canopy and interference scenarios. For instance, in dense urban canyons, signals drop in and out constantly (Soloviev et al., 2007), and a traditional tracking loop generally fails to lock onto a signal that is only available over several seconds or less. In contrast, an open loop receiver detects the signal as soon as it (re)appears, by identifying the energy peak in the 3D signal function, and immediately estimates signal parameters from the peak identified. To optimize the computational efficiency, the open-loop tracking is implemented utilizing the coarse zoom/fine zoom strategy (van Grass at al., 2009). In this case, signal energy peak is searched using a relatively coarse resolution of the 3D GPS signal function. Commonly used correlators,

with early, prompt and late code replicas and inphase and quadrature carrier components, are then placed around at the peak value to refine the resolution of signal parameters.

Another key feature of the deeply integrated architecture is that it supports independent tracking of individual satellite channels. This is illustrated in Figure 2.2.

Satellite signals are accumulated independently for each tracking channel over the entire accumulation period. Signal accumulation results are utilized by open-loop discriminators to compute adjustments to signal parameters that are generated by numerically controlled oscillators (NCOs). For example, adjustment to the carrier phase is calculated based on a four-quadrant arctangent function of the inphase (I) and quadrature (Q) accumulation results (van Graas et al., 2009). GPS measurements are then derived from NCO signal parameters for each satellite channel that is being tracked. This approach is different from the vector-tracking implementation (Bhattacharyya and Gebre-Egziabher, 2010-2011), where signal accumulation results are generally sampled at a 20-ms accumulation point and are then fed into the joint filter that computes the overall navigation and clock solution. Independent signal tracking for individual satellites maintains stochastically independent signal measurements for different satellite channels. The main benefit is that these measurements can be exploited for data quality monitoring, for example, by using Receiver Autonomous Integrity Monitoring (RAIM) techniques (Grover Brown, 1998), or GPS/INS integrity check (Farrell, 2006), which is especially beneficial in GPS-challenged environments such as dense forestry areas where large outliers can be present due to multipath errors. The measurement quality monitoring approach that was incorporated into the deeply integrated architecture is discussed later in this section.

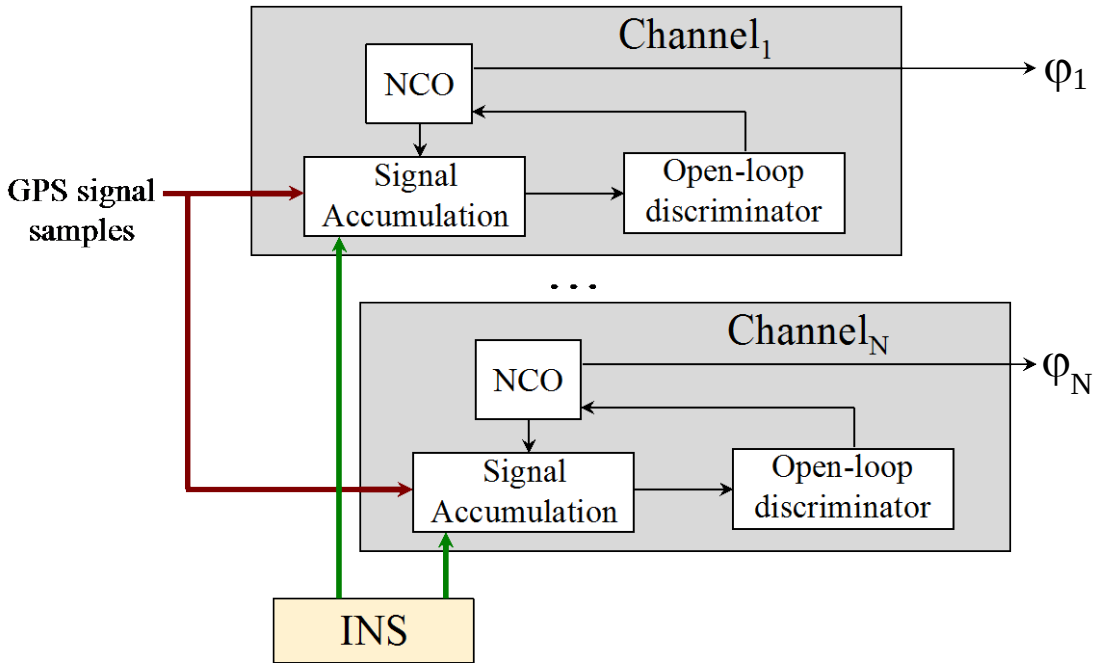



**Figure 2.2** Independent tracking of individual satellite channels

To enable long coherent accumulation of GPS signals and subsequent carrier phase tracking, inertial navigation must provide a reference dynamic trajectory that is accurate at a centimeter-per-second (cm/s) level. This is achieved by utilizing carrier phase measurements for the in-flight INS drift calibration. Specifically, the Kalman filter that estimates INS drift terms (see Figure 2.1 above) exploits GPS carrier phase as its measurement observable. One of the key aspects of the filter design is the use of carrier phase in degraded GPS environments. In such environments, the total number of satellites is generally insufficient to enable phase ambiguity resolution. Yet, the use of sub-centimeter accurate phase measurements is extremely beneficial, especially, for integration with lower-quality inertial units. Therefore, integer ambiguities are eliminated by differencing carrier phase over time and applying temporal phase differences as filter measurements. This approach is referred to as the dynamic-state INS calibration (Farrell, 2002; Wendel et al., 2006). It observes projections of position changes (instead of absolute position) on platform-to-satellite line-of-sight (LOS) and estimates the rest of inertial error states, including velocity errors, attitude errors and sensor biases, from these observations.

## 2.2 The Measurement Model

GPS observables of the Kalman filter for the satellite j are formulated as follows:

$$\Delta\tilde{r}_{GPS_j}(T_m) = -\frac{\lambda_{L_1}}{2\pi}\left(\tilde{\varphi}_{acm_j}(T_m) - \tilde{\varphi}_{acm_j}(T_{m-1})\right) + \left(\mathbf{e}_j(T_m), \mathbf{R}_{SV_j}(T_m)\right) - \left(\mathbf{e}_j(T_{m-1}), \mathbf{R}_{SV_j}(T_{m-1})\right) - \left(\mathbf{R}_{rcvr}(T_{m-1}), \mathbf{e}_j(T_m) - \mathbf{e}_j(T_{m-1})\right) \quad (2.1)$$

Where $\tilde{\varphi}_{acm_j}(T_m)$ and $\tilde{\varphi}_{acm_j}(T_{m-1})$ are carrier phase measurements for the current and previous accumulation intervals, respectively; $\lambda_{L_1}$ is the carrier wavelength; (. , .) is the vector dot product; $\mathbf{e}_j$ is the satellite/receiver LOS unit vector; $\mathbf{R}_{SV_j}$ is the satellite position vector; $\mathbf{R}_{rcvr}$ is the receiver position vector; and, the tilde sign indicates the presence of errors (primarily noise and multipath) in temporal range changes that are computed from carrier phase measurements. Note that carrier phase changes in Equation (2.1) are compensated for the satellite motion along the LOS (the second and third terms in the right-hand side of the equation) and for changes in relative satellite/receiver geometry (the forth term in the right-hand side of the equation). Hence, only the receiver motion term remains in the GPS observable of the Kalman filter.

In Equation (2.1), the satellite position vector $\mathbf{R}_{SV_j}$ is calculated based on ephemeris data, while the receiver position vector $\mathbf{R}_{rcvr}(T_{m-1})$ is estimated based on code phase measurements obtained at the previous measurement epoch (i.e., for the previous signal integration interval). The LOS unit vector **e** is computed as $\mathbf{e}_j = \left(\mathbf{R}_{SV_j} - \mathbf{R}_{rcvr}\right) \Big/ \left|\mathbf{R}_{SV_j} - \mathbf{R}_{rcvr}\right|$, where $|.|$ is the vector absolute value.

It can be shown [11] that $\Delta\tilde{r}_{GPS_j}$ is expressed as follows:

$$\Delta\tilde{r}_{GPS_j}(T_m) = -\left(\mathbf{e}_j(T_m), \Delta\mathbf{R}(T_m)\right) + \Delta\delta t_{rcvr}(T_m) + \varepsilon_j \quad (2.2)$$

where $\Delta\mathbf{R}$ is the position change between GPS updates; $\Delta\delta t_{rcvr}$ is the accumulated clock drift; and, $\varepsilon_j$ is the carrier phase error term that combines thermal noise and multipath error components. The analysis presented in (van Graas and Soloviev, 2004) indicates that errors in temporally differenced ephemeris and atmospheric delays generally stay below the noise and multipath error level. Therefore, these errors are not included into consideration. The Equation (2.3) defines inertial observables of the Kalman filter:

$$\Delta\tilde{r}_{INS_j}(T_m) = \left(-\mathbf{e}_j(T_m), \Delta\tilde{\mathbf{R}}(T_m) + \left(\tilde{\mathbf{C}}_b^N(T_m) - \tilde{\mathbf{C}}_b^N(T_{m-1})\right)\cdot \boldsymbol{L}_b\right) \tag{2.3}$$

In Equation (2.3):

$\Delta\tilde{\mathbf{R}}$ is the INS-based position change between GPS updates;

$\tilde{\mathbf{C}}_b^N$ is the INS-computed direction cosine matrix for the coordinate transformation from the body (b) frame to the navigation (N) frame); and,

$\boldsymbol{L}_b$ is the lever arm vector from the inertial measurement unit to the GPS receiver with vector components being resolved in the body-frame.

The filter measurement observation vector is defined as a difference between inertial and GPS observables:

$$\mathbf{y}_{Kalman}(T_m) = \begin{bmatrix} \Delta\tilde{r}_{INS_1}(T_m) - \Delta\tilde{r}_{GPS_1}(T_m) \\ \cdots \\ \Delta\tilde{r}_{INS_{N_{SV}}}(T_m) - \Delta\tilde{r}_{GPS_{N_{SV}}}(T_m) \end{bmatrix} \tag{2.4}$$

where $N_{SV}$ is the total number of visible satellites. The filter states include:

- INS error states: 18 states total, including: delta position errors, $\delta\Delta\mathbf{R}$, for the error in position change between consecutive GPS update epochs (3 states); velocity error states $\delta\mathbf{V}$ (3 states); attitude error states $\delta\boldsymbol{\theta}$ (3 states); delta attitude error states, $\delta\Delta\boldsymbol{\theta}$, for the change in the attitude

errors between carrier phase measurement epochs (3 states); sensor bias states $\mathbf{d}_{gyro}$ and $\mathbf{b}_{accel}$ (3 states for gyros and 3 states for accelerometers);

- GPS receiver clock states: receiver clock drift accumulated between consecutive carrier phase updates $\Delta\delta t_{rcvr}$ ; and, receiver clock drift $\delta\dot{t}_{rcvr}$.

Note that the Kalman filter is formulated in the complementary form and estimates INS errors in navigation states rather than estimating the states themselves. The extended Kalman filter (EKF) formulation is applied. Accordingly, linearized Kalman filter measurements are derived by applying the first-order Taylor series expansion to observation equations (2.3) through (2.5). For the filter state vector that includes the states in the order defined above, the row j of the Kalman filter measurement observation matrix, $\mathbf{H}_{Kalman}$, that corresponds to the satellite j observation is derived as:

$$\mathbf{H}_{\text{Kalman}_j} = \left[ -\mathbf{e}_j^T \quad \mathbf{0}_{1\times 3} \quad \mathbf{e}_j^T \cdot \left[ \left( \tilde{\mathbf{C}}_b^N(T_m) - \tilde{\mathbf{C}}_b^N(T_{m-1}) \right) \cdot \boldsymbol{L}_b \right]_\times \quad -\mathbf{e}_j^T \cdot \left[ \tilde{\mathbf{C}}_b^N(T_{m-1}) \cdot \boldsymbol{L}_b \right]_\times \quad \mathbf{0}_{1\times 3} \quad \mathbf{0}_{1\times 3} \quad 1 \quad 0 \right] \qquad (2.5)$$

where $\mathbf{0}_{1\times 3}$ is the one-by-three zero row; and, $\times$ denotes the skew-symmetric matrix. The measurement covariance is setup up as a diagonal matrix (i.e., measurement errors for different satellites are assumed to be uncorrelated), with the diagonal terms defined by the standard deviation of carrier phase error (that is set at a sub-cm level). With the measurement observation matrix defined by equation (4), the Kalman filter implements its estimation routine using a standard strapdown INS error propagation mechanism [9] to implement the prediction update.

As mentioned previously, measurement quality monitoring is applied to remove outliers. Figure 2.3 illustrates the algorithmic solution.

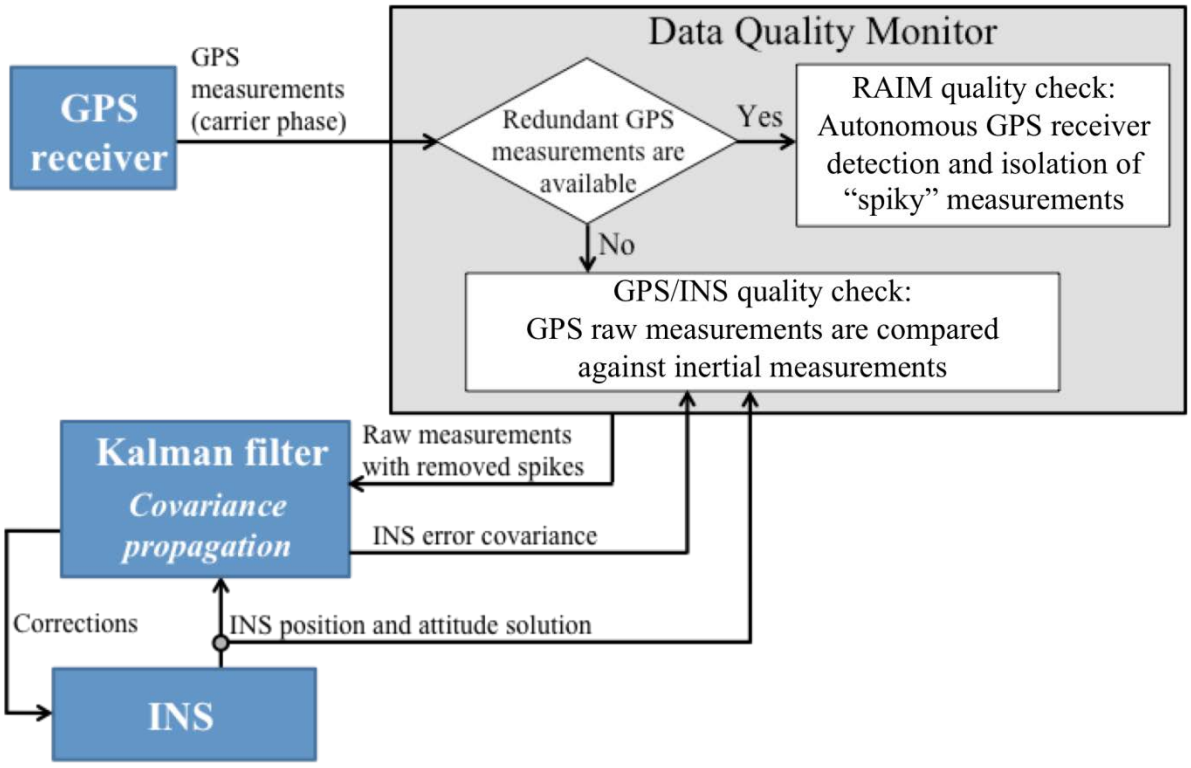



**Figure 2.3** Measurement quality monitoring for detection and identification of outliers

The algorithm first attempts to implement an internal GPS quality check that is based on RAIM. The RAIM-based check is applied for cases where the number of available satellites is five or more. If the internal quality check detects the presence of a failure or insufficient satellites (less than five) are available to execute RAIM, then INS-based quality check is also implemented. INS-based quality check predicts GPS measurements using inertial data, compares predicted measurements with actual measurements and removes those measurements for which large discrepancies between predicted and measured values are found.

The failure detection is based on the QR-factorization parity check of the least mean square (LMS) estimation algorithm that computes position change from temporal changes in carrier phase. The computations are described in step-by-step details in reference (van Graas and Soloviev, 2004). Temporal carrier phase changes are computed and adjusted for satellite Doppler and geometry terms (as formulated by Equation (2.1) above). Adjusted carrier phase changes are then applied to formulate observations for the LMS estimation:

$$\mathbf{H}\cdot\begin{bmatrix}\Delta\mathbf{R}\\ \Delta\delta t_{rcvr}\end{bmatrix}=\begin{bmatrix}\Delta r_{GPS_1}\\ \dots\\ \Delta r_{GPS_{N_{SV}}}\end{bmatrix}+\begin{bmatrix}\varepsilon_1\\ \dots\\ \varepsilon_{N_{SV}}\end{bmatrix},\ \mathbf{H}=\begin{bmatrix}\mathbf{-e}_1 & 1\\ \dots & \dots\\ \mathbf{-e}_{N_{SV}} & 1\end{bmatrix} \tag{2.6}$$

where $N_{SV}$ is the number of available satellites. QR-decomposition of the observation matrix **H** yields:

$$\mathbf{H}=\mathbf{q}\cdot\mathbf{r} \tag{2.7}$$

where **q** is an orthonormal matrix (i.e., $\mathbf{q}^{-1}=\mathbf{q}^{T}$) and ***r*** is an upper triangular matrix, i.e. $\mathbf{r}=\begin{bmatrix}\mathbf{U}\\ \mathbf{0}\end{bmatrix}$.

For the case of position change (3 components) and clock drift estimation, **U** is an $4\times4$ upper triangular matrix and **0** is a $(M-4)\times4$ zero matrix. Note that $M>4$ is required to perform QR-decomposition, i.e. at least one redundant measurement is needed. Based on Equations (2.5) and (2.6) and using the orthonormal matrix property:

$$\mathbf{q}^{T}\cdot\begin{bmatrix}\Delta r_{GPS_1}\\ \dots\\ \Delta r_{GPS_{N_{SV}}}\end{bmatrix}=\mathbf{r}\cdot\begin{bmatrix}\Delta\mathbf{R}\\ \Delta\delta t_{rcvr}\end{bmatrix}-\mathbf{q}^{T}\cdot\begin{bmatrix}\varepsilon_1\\ \dots\\ \varepsilon_{N_{SV}}\end{bmatrix} \tag{2.8}$$

The transpose of the **q** matrix is partitioned as follows:

$$\mathbf{q}^{T}=\begin{bmatrix}\mathbf{q}_U\\ \mathbf{q}_p\end{bmatrix} \tag{2.9}$$

where $\mathbf{q}_U$ and $\mathbf{q}_p$ are $4\times M$ and $(M-4)\times M$ matrices, respectively. From Equations (2.8) and (2.9) it follows that

$$\begin{aligned}\mathbf{q}_U\cdot\begin{bmatrix}\Delta r_{GPS_1}\\ \dots\\ \Delta r_{GPS_{N_{SV}}}\end{bmatrix}&=\mathbf{U}\cdot\begin{bmatrix}\Delta\mathbf{R}\\ \Delta\delta t_{rcvr}\end{bmatrix}+\mathbf{q}_U\cdot\begin{bmatrix}\varepsilon_1\\ \dots\\ \varepsilon_{N_{SV}}\end{bmatrix}\\ \mathbf{q}_p\cdot\begin{bmatrix}\Delta r_{GPS_1}\\ \dots\\ \Delta r_{GPS_{N_{SV}}}\end{bmatrix}&=-\mathbf{q}^{(p)}\cdot\begin{bmatrix}\varepsilon_1\\ \dots\\ \varepsilon_{N_{SV}}\end{bmatrix}\end{aligned} \tag{2.10}$$

The first part of equation (9) is applied for the LMS solution. The second part provides parity residuals:

$$\mathbf{p} = \mathbf{q}_p \cdot \begin{bmatrix} \Delta r_{GPS_1} \\ \dots \\ \Delta r_{GPS_{N_{SV}}} \end{bmatrix} = -\mathbf{q}^{(p)} \cdot \begin{bmatrix} \varepsilon_1 \\ \dots \\ \varepsilon_{N_{SV}} \end{bmatrix} \tag{2.11}$$

The residual vector **p** determines the projection of the error vector into the LMS solution null space. The parity residual test compares parity elements with elements of a threshold vector $\gamma$ that is defined by 3-sigma values of carrier phase noise that is transformed into the parity space, i.e.:

$$\begin{aligned} &\gamma = [\gamma_k],\ \gamma_k = 3\sqrt{\alpha_k} \\ &\alpha = \mathbf{q}^{(p)} \left(\mathbf{q}^{(p)}\right)^T \sigma_\varepsilon^2 \end{aligned} \tag{2.12}$$

where $\sigma_\varepsilon^2$ is the standard deviation of error in carrier phase temporal differences. Note that we do not explicitly assume a single satellite failure case that is used by GPS RAIM techniques to define failure protection levels of the integrity monitoring procedure. An exceeded threshold simply means that measurement failures can be present and the quality monitoring needs to further check the measurements using the GPS/inertial approach that is described next.

If the RAIM-based quality check identifies the presence of outliers or if the measurement availability is insufficient to support the RAIM, the INS-based quality check is further used to verify the measurements (if necessary, i.e. for cases of limited satellite availability) and isolate the failed measurement channels. The quality check procedure predicted and measured values of adjusted phase differences, i.e.:

$$\begin{aligned} &\left|\Delta\tilde{r}_{GPS_j} - \Delta\tilde{r}_{INS_j} - \Delta\delta\hat{t}_{rcvr}\right| \le 3\sigma_j \rightarrow \text{fault} - \text{free} \\ &\text{otherwise, failure is detected} \\ &j = 1,\dots,N_{SV} \end{aligned} \tag{2.13}$$

where $\Delta\delta\hat{t}_{rcvr}$ is the Kalman filter estimate of the receiver clock drift that is accumulated between GPS updates; and, $\sigma_j$ is the sigma value of the prediction/measurement difference for the fault-

free case. This value is computed based on the INS error covariance and standard deviation of the carrier phase error:

$$\sigma_j^2 = \mathbf{H}_{\text{Kalman}_j} \cdot \mathbf{P} \cdot \mathbf{H}_{\text{Kalman}_j}^{\text{T}} + 2\sigma_{\varepsilon_j}^2 \tag{2.14}$$

where **P** is the Kalman filter state covariance matrix and $\sigma_j$ is the standard deviation of the carrier phase measurement error. The factor of two multiplication is applied herein since temporal differences in carrier phase are utilized for GPS measurements.

# 3. TEST AREA

Figure 3.1 shows a picture of the test setup that was used to acquire experimental data.

(a) (b)

**Figure 3.1** Test setup; a: OSU GPSVan, b: OU GPS software receiver

Equipment was mounted in the OSU GPSVan vehicle that is shown in Figure 3.1a. The data collection suite includes:

- NovAtel OEM-V GPS receiver that provides GPS signal measurements (pseudo-ranges and carrier phase);
- Instrumentation-grade radio-frequency (RF) front-end (Ohio University Transform-domain Instrumentation GNSS Receiver, TRIGR (Gunawardena et al., 2007)) and data collection server that were used to acquire and store raw GPS signal samples, i.e. GPS signal that is down-sampled to the baseband. The front-end and server are shown in Figure 3.1b;
- GPS antennas; and,
- Honeywell H764G navigation-grade inertial measurement unit (IMU) with the stand-alone position drift of 0.8 nmi/hr.

In order to evaluate the system performance for lower-grade IMU options, tactical-grade and lower-grade inertial data were emulated by adding simulated errors (including gyro drift, gyro noise, accelerometer bias and accelerometer noise) to experimental data recorded by the navigation-grade inertial unit. Additional gyro-drift for emulated tactical-grade data was simulated as a first order Gauss-Markov process with a one-sigma value of 10 deg/hr and a 1 hr time correlation interval. Simulated gyro noise corresponds to the random walk of 4 $\text{deg}/\sqrt{\text{hr}}$. Accelerometer bias was simulated as a first-order Gauss Markov process with the standard deviation of 0.1 mg and 1 hr correlation time; 0.1 $\text{mg}/\text{s}^2$ accelerometer noise was also simulated.

Test data were collected in suburban tree-covered environments of Columbus, Ohio and in dense forestry areas of Wayne National Forest in Athens County, Ohio. Figure 3.2 shows an example data collection environment in the Wayne National Forest.

**Figure 3.2** Example test environment: dense canopy scenario in the Wayne National Forest, Ohio

The images show dense canopy coverage where only very limited portions of clear sky are available.

Experimental data were post-processed by three solution options that include:

1) Kinematic GPS position solution that was obtained by post-processing NovAtel receiver measurements using Waypoint® software products;
2) Tightly-coupled GPS/INS software package developed by the Satellite Positioning and Inertial Navigation (SPIN) Laboratory at The Ohio State University; this package exploits the tightly coupled integration approach that uses carrier phase measurements with resolved integer ambiguities for the inertial drift estimation; and,
3) Deeply integrated GPS/INS architecture.

Evaluation of positioning precision was performed as follows. Precise reference trajectory that is based on kinematic carrier phase GPS solution is not available in dense forestry areas. As a result, direct precision evaluation cannot be performed. Therefore, we considered three alternative options:

1) *Consistency of the reconstructed test trajectory*: This option uses consistency of the motion trajectory to validate the positioning precision. This includes, displaying the trajectory in Google Earth, or in any orthoimagery available, and validating that it closely follows the road; comparing forward and return paths and demonstrating consistency of the results.
2) *Attenuation of open-sky GPS signals to emulate forestry data*: Analysis of forestry data allows evaluation of the signal attenuation factor that is applied to GPS signals during the propagation through the canopy. This signal attenuation is artificially injected to open-sky GPS signals. Attenuated GPS signals are then processed by the deeply integrated solution and results are compared to the reference kinematic GPS trajectory that is available in open-sky environments. Therefore, for the second precision evaluation option we implemented the following steps. *First,* GPS signal attenuation in dense forestry areas was analyzed. *Second*, open-sky test segments with reliable kinematic reference trajectory were chosen. *Third*, signal attenuation that closely resembles the attenuation

due to canopy was added to open-sky GPS signals (by adding simulated broadband noise). *Forth*, attenuated signals were processed by the deeply integrated architecture to compute the position solution. Finally, *fifth*, the deep integration solution was compared to the reference trajectory.

3) *Consistency of positioning results between multiple test trials*: For this type of evaluation, experimental data were collected for multiple trajectories driven along the same path. Position estimation results for different test trials were then compared to each other.

# 4. PERFORMANCE ANALYSIS

Initial data collects were organized in Columbus, Ohio, and then the major data acquisition campaign was performed in the Wayne National Forest, Athens County, Ohio.

## 4.1 Columbus Data Analysis

Figure 4.1 shows example test results for a tree-covered suburban area in Columbus.

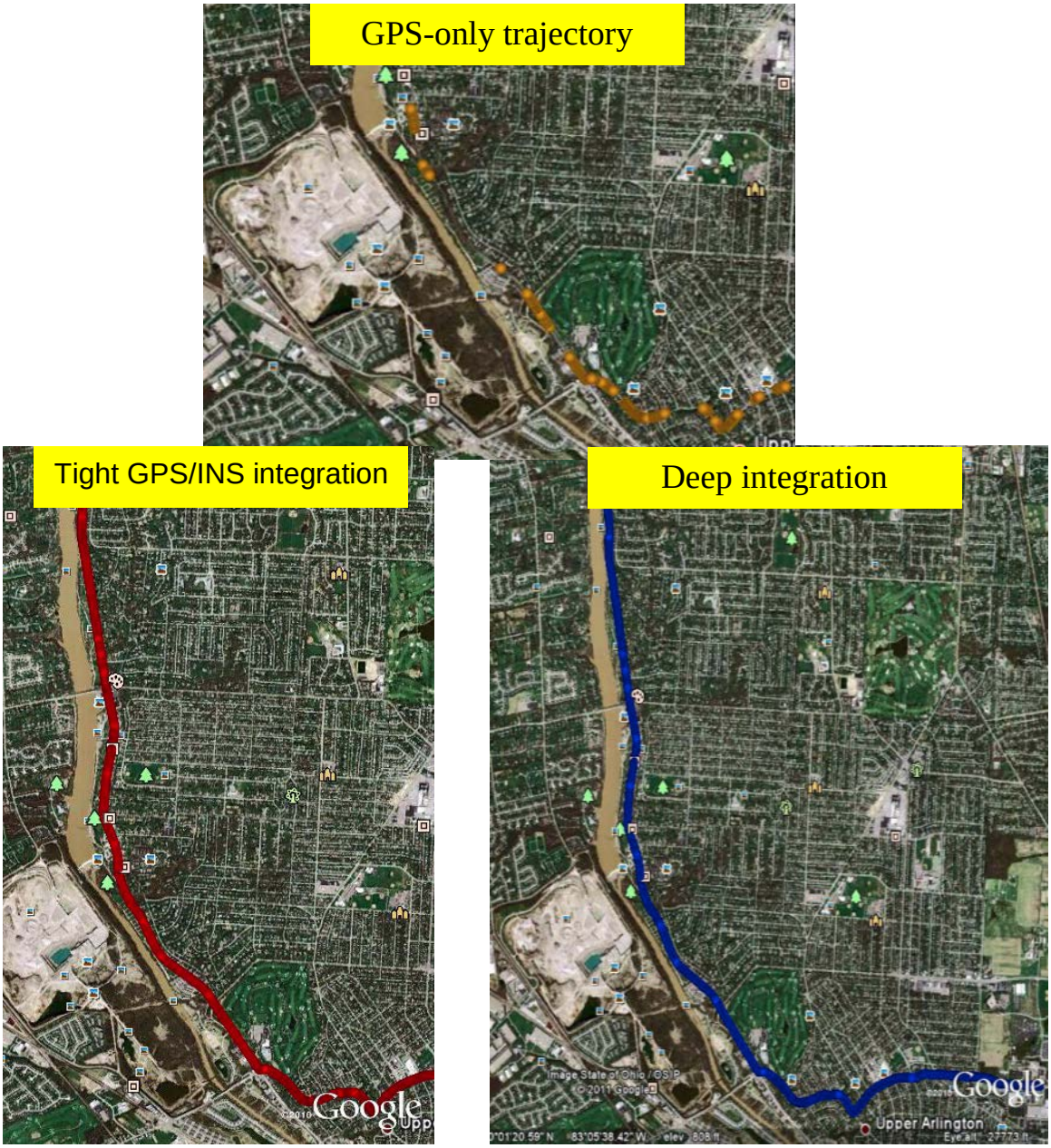



**Figure 4.1** Example test results: tree-covered suburban areas of Columbus, OH

Kinematic GPS, tightly-coupled GPS/INS and deeply integrated GPS/INS position trajectories are plotted by entering longitude and latitude estimates of the navigation solution into Google Earth. Tight coupling and deep integration solutions are shown for the case of navigation-grade inertial unit. The kinematic GPS solution can maintain position estimation for those portions of the test path where unobstructed sky view is available. As a result, the solution is extremely sparse and supports the required navigation capabilities for a very limited portion of the test. In contrast, both tightly-coupled and deeply integrated approaches support the trajectory reconstruction for the entire duration of the test. For the majority of the test route, tight integration and deep integration produce similar results. However, for some limited portions of the test where increased tree coverage was present, the use of deep integration improves the solution precision. This is exemplified in Figure 4.2.

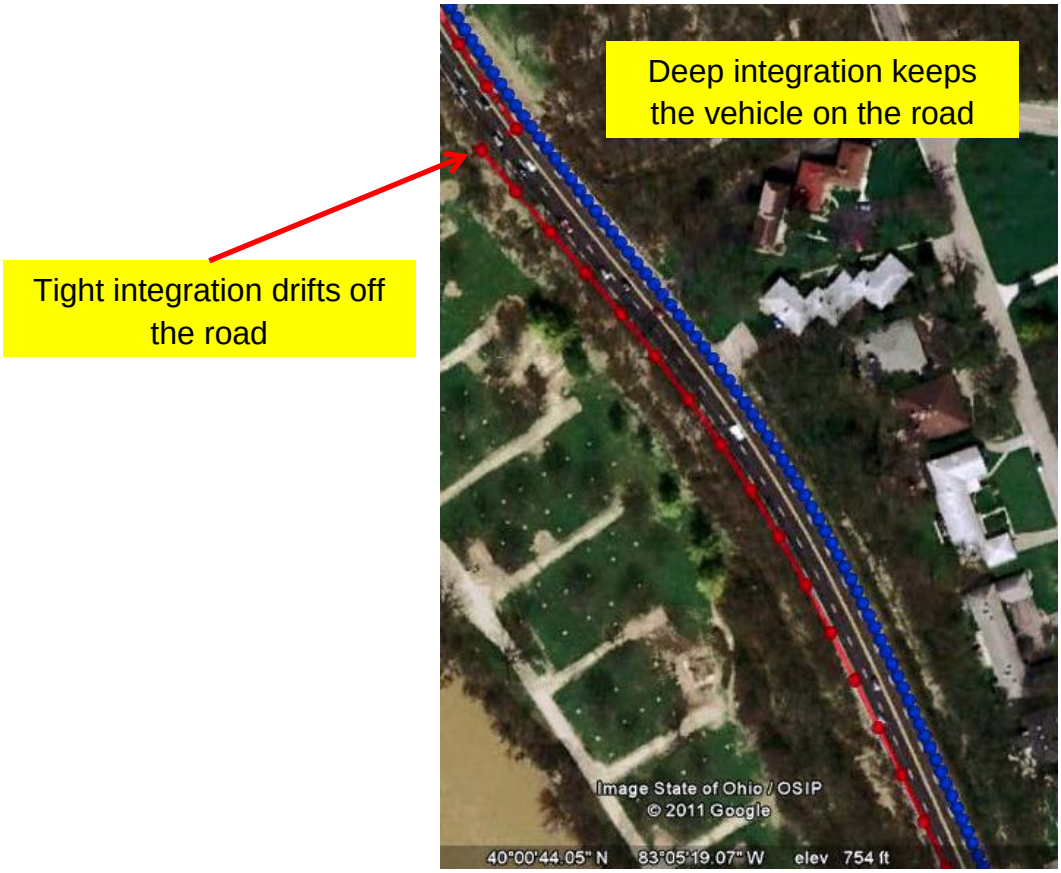



**Figure 4.2** Improvement in position precision achieved by deep integration

In this example, tightly coupled solution drifts off the road (the maximum lateral position error is 15.4 meters), while deep integration maintains position estimates within the correct road lane that was driven during the test.

## 4.2 Wayne National Forest Data Analysis

Figure 4.3 shows the kinematic GPS solution for the example test scenario in the Wayne National Forest. Similar to the Columbus test results, the GPS-only option maintains position estimates only when the test trajectory crosses over open-sky areas, which severely limits trajectory reconstruction capabilities.

**Figure 4.3** Wayne National Forest test example: Kinematic GPS solution

Figure 4.4 shows example test results for the tight integration option that uses the navigation-grade inertial unit. The results presented demonstrate that tight integration supports precise positioning capabilities for limited segments of the test trajectory, while large solution discontinuities can be present for the other parts of the test route.

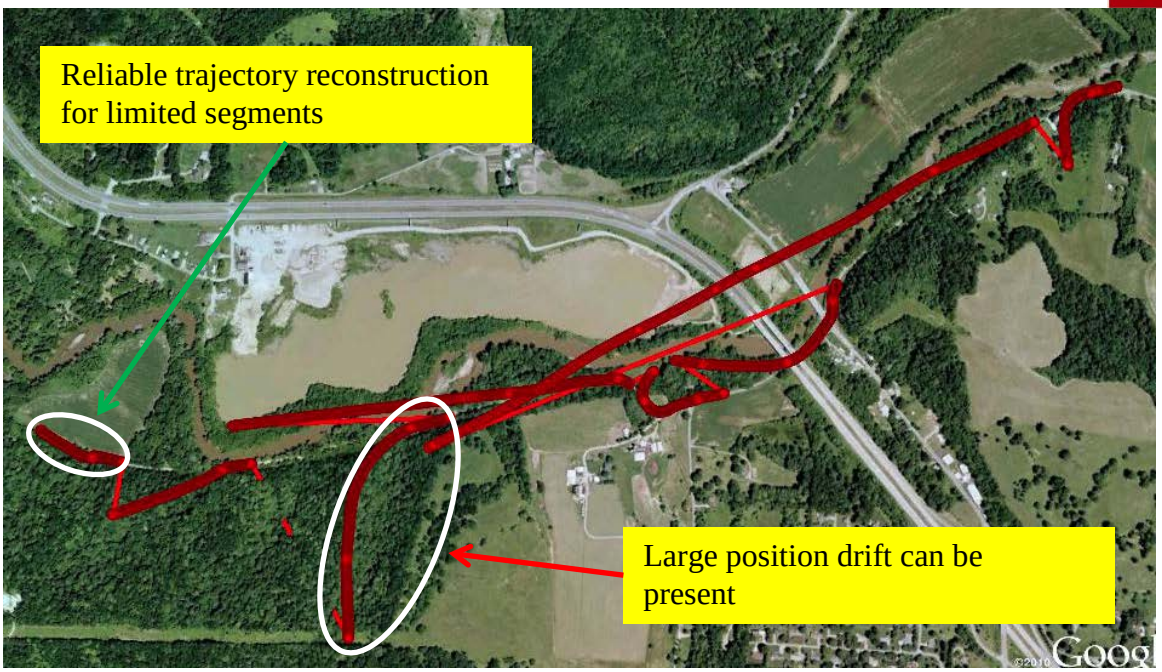



**Figure 4.4** Wayne National Forest test example: Position solution of tight integration with the navigation-grade IMU

Figure 4.5 shows example deep integration test results. The solution is presented for the case where GPS signal samples are combined with navigation-grade IMU data. As shown in Figure 4.5, deep integration enables reliable trajectory reconstruction for the entire test route. No discontinuities are present in the deeply-integrated position solution. Forward path and return path are closely spaced and distinguishable, which corresponds to the actual path driven where the forward and return trajectories were generally separated by 1-2 m. The reconstructed trajectory closely follows the actual test route (a country road that was driven by the test vehicle) both for relatively open areas and forestry areas with extreme foliage density. At the end of the test loop, the reconstructed trajectory returns to the starting point.

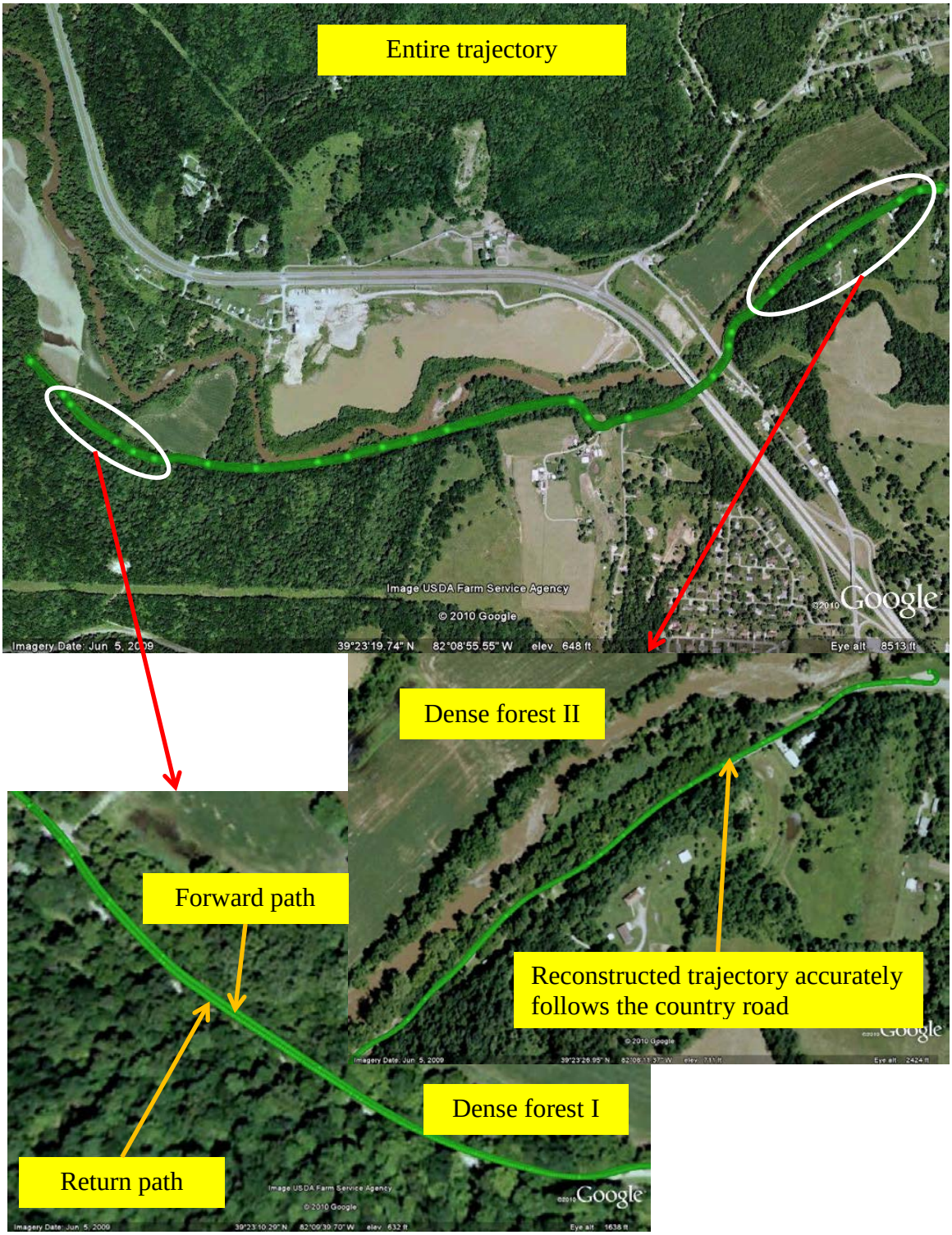



**Figure 4.5** Wayne National Forest test example: Position solution of the deeply integrated GPS/inertial with the navigation-grade IMU

Figure 4.6 shows example simulation results for the case where emulated tactical-grade inertial measurements were applied for the deep integration. A tactical-grade IMU was emulated as 10 deg/hr (1 sigma value) gyro drift and 3 deg/sqrt(hr) (1 sigma value) gyro noise; and 1 milli-g (mg) (1 sigma value) accelerometer bias and 0.01 m/s2 (1 sigma value) accelerometer noise.


**Figure 4.6** Wayne National Forest test example: Position solution reconstructed by deep integration that uses the tactical-grade IMU

As shown in Figure 4.6, deep integration still maintains a reliable trajectory reconstruction when the navigation-grade inertial unit is substituted by the tactical grade option.

For lower-grade IMUs (emulated as 100 deg/hr gyro drift and 30 deg/srt(hr) gyro noise; 2 mg accelerometer bias and 2 m/s2 accelerometer noise), it was found that inertial drift performance does not allow for cm/s accurate aiding of the GPS signal accumulation. As a result, deep integration is not able to support robust trajectory reconstruction capabilities in dense forestry areas.

We have also investigated sensitivity of the deeply integration solution to the use of data quality monitoring algorithm and reduced signal integration time. Figure 4.7 shows the test results for the case where the tactical-grade inertial data is used and the data quality control is not implemented. The results clearly show that the trajectory reconstruction becomes unreliable in this case. Therefore, it is crucial to detect measurement outliers and exclude them from the EKF-based navigation solution.

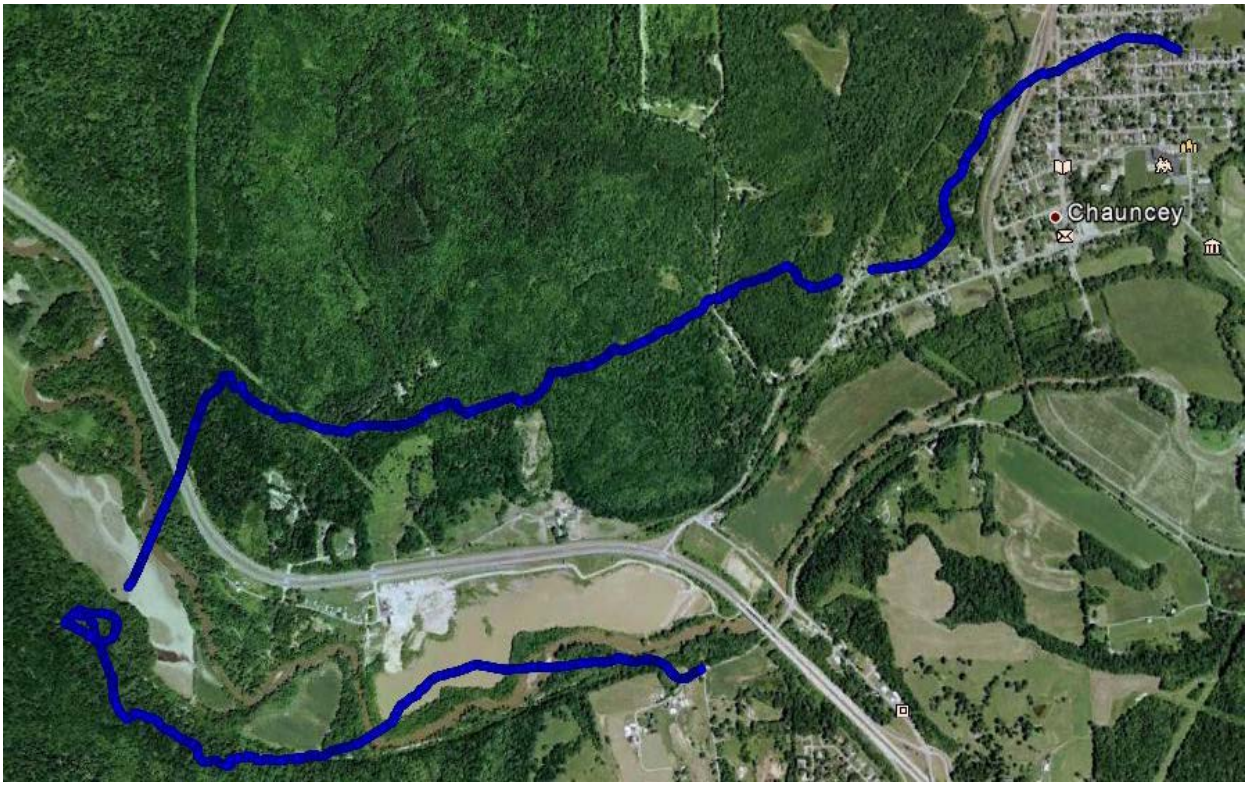



**Figure 4.7** Wayne National Forest test example: Trajectory reconstruction results without measurement quality control

Figure 4.8 shows test results for the tactical-grade IMU and the case where the signal accumulation interval is reduced to 20 ms; i.e., an unaided open-loop GPS receiver tracking is applied and the deep integration option is not used.

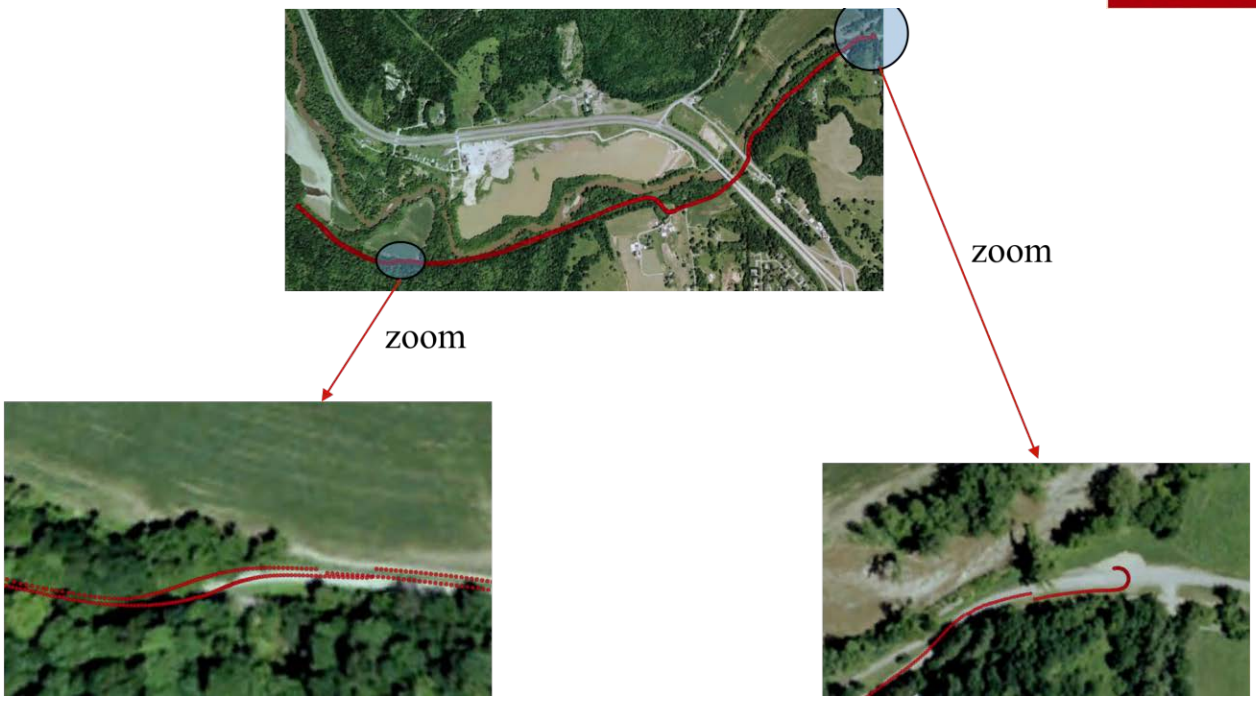



**Figure 4.8** Wayne National Forest test example: Trajectory reconstruction results with reduced (20-ms) signal accumulation

Overall, a smooth trajectory reconstruction is maintained. However, for select portions of the trajectory the reconstructed solution deviates from the road and solution jumps can be present. Specifically, the jump value in the left-hand zoomed image is about 3.8 m, the jump in the right hand zoomed-image is 6 m, approximately. Hence, the decrease of the signal integration interval still maintains a continuous trajectory reconstruction, but decreases the solution precision to the level of a few meters.

As mentioned previously, GPS forestry signals were emulated for open-sky portions of test trajectories, where the kinematic GPS solution is available and can be used as a reference, in order to evaluate the absolute precision performance of the deeply integrated solution. Figure 4.9 shows typical C/No estimates recorded in dense forestry areas.

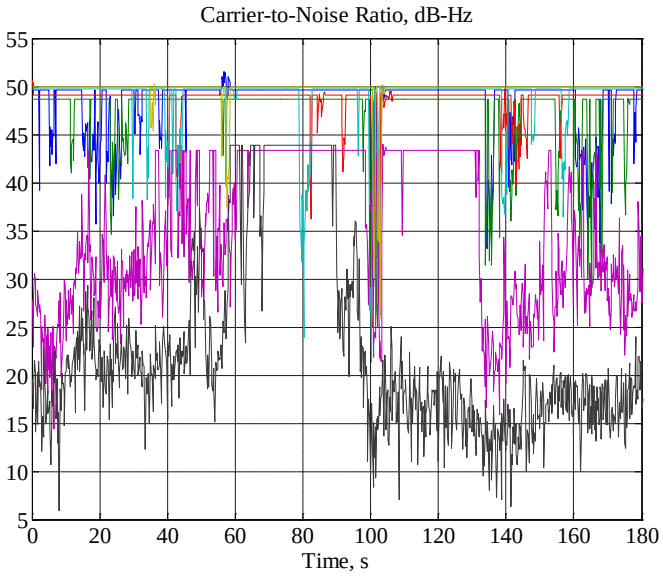



**Figure 4.9** C/No estimates in forestry environments, Wayne National Forest Test (different colors correspond to different satellites)

Accordingly, simulated broad-band noise was added to open-sky GPS signal samples, such that the resultant C/No ratio corresponds to C/No values that were observed under dense canopy. Figure 4.10 shows an example test trajectory where “canopy-like” attenuation was applied to (relatively) open-sky GPS signals. The overall duration of the test was approximately three minutes. This was the longest interval where the clear sky data was available.


**Figure 4.10** Example open-sky test trajectory applied for the precision evaluation of the deeply integrated GPS/INS solution

Figure 4.11 shows precision evaluation results for navigation-grade and tactical-grade IMUs. The position residuals herein are computed as the difference between the deep integration position estimates (derived from artificially attenuated GPS signals) and position estimates of the kinematic GPS solution that was computed from the open-sky GPS signal measurements.

The position precision is primarily limited by the un-calibrated heading error that cannot be accurately estimated due to the lack of vehicle maneuvers. As a result, navigation-grade and tactical-grade inertial integration options exhibit similar precision performance. The example plots presented demonstrate sub-meter positioning precision (the maximum error is about 0.4 meters). Overall, test results presented in this section demonstrate that the deep integration approach serves as a reliable option for enabling precise position capabilities in dense forestry areas.

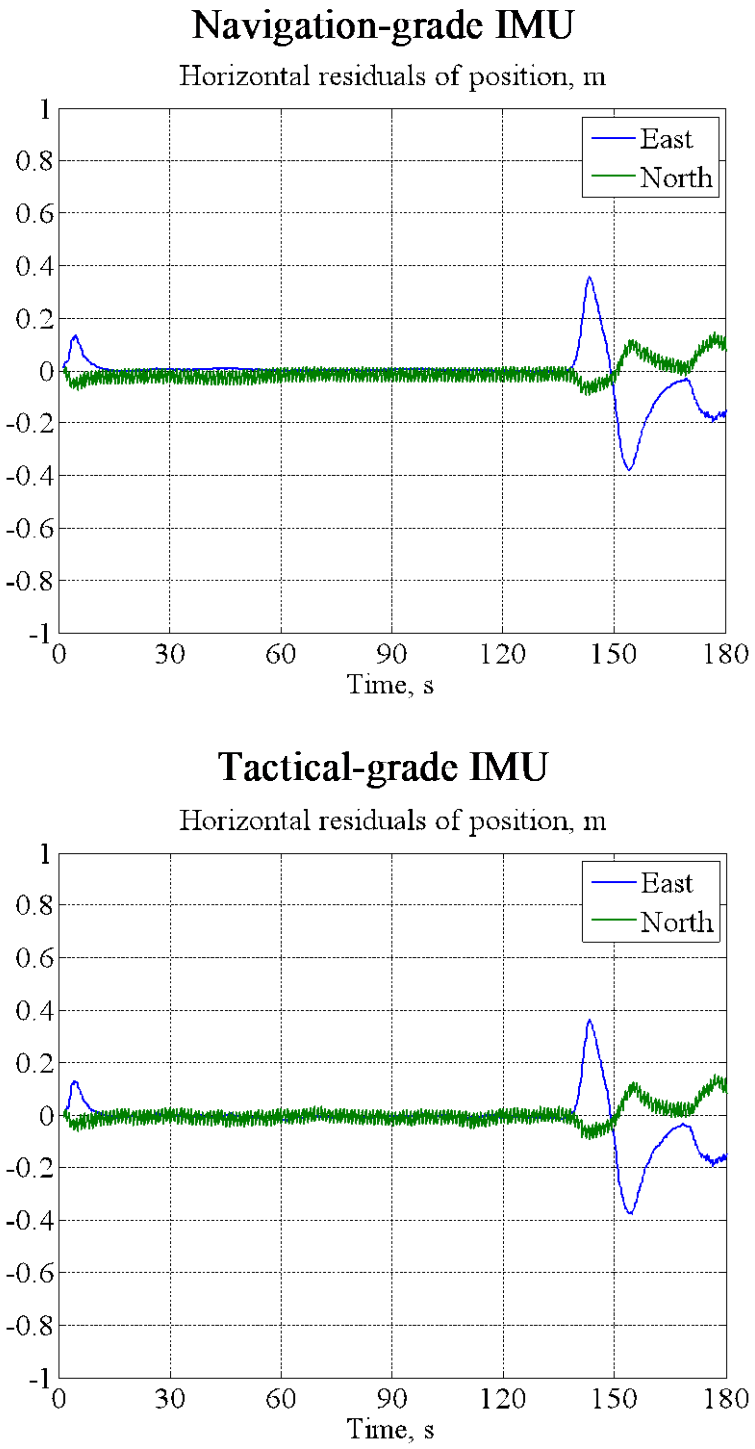



**Figure 4.11** Example precision evaluation results: Deep integration computed from artificially attenuated GPS signals vs. kinematic solution (derived from open-sky GPS data)

As it was stated previously, the final performance evaluation metrics includes comparing trajectory reconstruction results for multiple trials driven along the same path. These results demonstrate that a meter-level agreement between different independently reconstructed trajectories is generally achieved. Figures 4.12 through 4.15 show example results that include two test trajectories.

**Figure 4.12** Trajectory reconstruction consistency for multiple tests: Google Earth representation of deep integration position estimation results; green and blue dots correspond to two independent test trajectories driven along the same path


**Figure 4.13** Trajectory reconstruction consistency for multiple tests: Google Earth representation of deep integration position estimation results; green and blue dots correspond to two independent test trajectories driven along the same path; zooming on select test segments

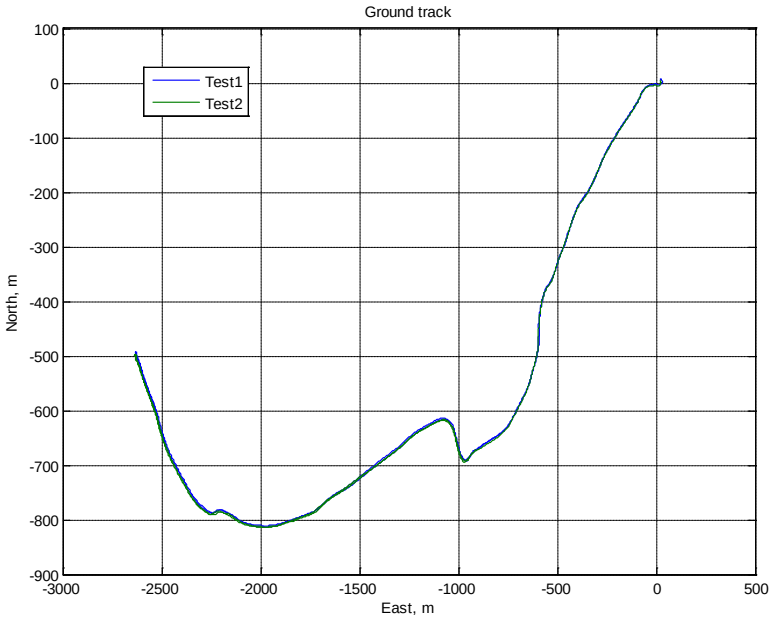



**Figure 4.14** Consistency of trajectory reconstruction for multiple tests: local ENU representation of deep integration position estimation results

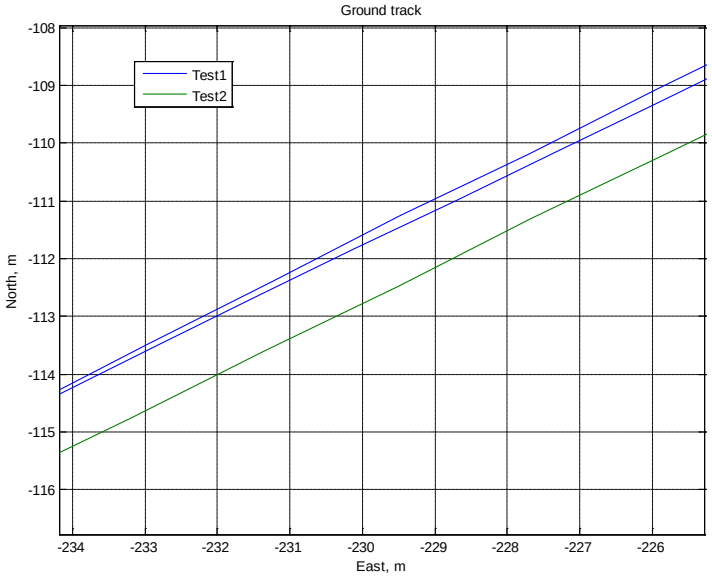



**Figure 4.15** Consistency of trajectory reconstruction for multiple tests: local ENU representation of deep integration position estimation result; zooming on a select portion of the test trajectory

# 5. GEOLOCATION NAVIGATION SOFTWARE MODULES

## 5.1 AIMS PRO QUADRUPLE INTEGRATION NAVIGATION SOFTWARE

### 5.1.1 Introduction

The original AIMS PRO software package has been developed in Microsoft Visual Studio C++ environment. The development platform was chosen mainly for performance reasons, though many algorithmic components were originally implemented and tested in Matlab environment. However, as computer systems have improved in processing power and Matlab execution performance has been increased recently, the Matlab environment is quite adequate and there is no need to convert the code to VC++. In addition, the software receiver component is completely in Matlab implementation. Therefore, the AIMS PRO navigation software has been also fully implemented in Matlab in the second phase of the project, as it provides a better environment for further developments, testing, etc. Obviously, all the new functionalities developed during this project have been programmed only in Matlab.

This section shortly describes the software design and usage of the GPS/IMU based multi-sensor integration software module using the Matlab environment. The design and development of this Matlab based software module is to be served as the SDK (Software Development Kit) for further prototype development of GPS/IMU related multi-sensor integrated systems; for example, the GPS/IMU integration with a variety feedback from image sensors, such as LiDAR, flash LADAR, camera, etc. These extensions go beyond the spherical target based feedback, developed in the first phase of the project.

The major features of this GPS/IMU based multi-sensor integration software module include:

- GPS/IMU integration using an Extanded Kalman filter (EKF)
- 24-dimensional state vector to model the platform position (3), velocity (3), attitude/orientation (3), accelerometer bias (3), gyro bias (3), accelerometer scale factor error (3), gyro scale factor error (3), and GPS antenna level arm offset w.r.t IMU body center (3, optional, not needed if the level arm offset is accurately known or ignorable)
- Using psi-angle linearized inertial error model
- Using 6-state driven process noise model (3 accelerometer measurement noise and 3 gyro measurement noise)
- Autonomous stationary detection based on accelerometer and gyro outputs to trigger ZUPT (Zero velocity measurement UPdaTe)

- NON-Holonomic Constraint (NHC) for vehicle or personal navigation systems (pushcart or backpack sensor configurations)
- Ability to accommodate measurement update for the position, and orientation changes between two epochs
    - By augmenting two state vectors into one state vector
- Support loosely-coupled (LC) integration mode
    - Supporting position measurement update
    - Supporting velocity measurement update
- Support tightly-coupled (TC) integration mode (some components are still in testing phase)
    - Support GPS and GLONASS dual-frequency data processing
    - Support GNSS RINEX 2.1 data format
    - Support IGS SP3 precise ephemeris for GPS and GLONASS
    - Only support one reference and one rover stations
    - GNSS (GPS + GLONASS) single-difference code and carrier phase measurement update on both L1 and L2 frequencies
    - Additional state vector (number of single-differences between two receivers) to model the ionospheric delay residuals, modeling as random walk
    - Additional state vector (number of single-differences between two receivers) L1 ambiguities, modeling as random constant
    - Additional state vector (number of single-differences between two receivers) L2 ambiguities, modeling as random constant
    - Integer AR (Ambiguity Resolution) based on LAMBDA method

The following provides a basic description of the AIMS PRO extended Matlab implementation; the Kalman filtering details are omitted, and only the input/output and integration functionality are discussed.

### 5.1.2 System Settings

To control the navigation filtering process, the system parameters, inputs and outputs are interfaced by the function "`make_filter_option.m`"; details are shown in Table 1.

| Field | Default | Description |
|---|---|---|
| posFileName | '' | GPS Position (and Velocity) File Name |
| posFileType | 0 | GPS Poisition (and Velocity) File Type<br>0: Text File Format (Time, lat, lon, ht, RMS X, RMS Y, RMS Z, (Vx, Vy, Vz, RMS Vx, RMS Vy, RMS Vz)), note: Velocity is optional<br>1: Binary file format (Time, lat, lon, ht, RMS X, RMS Y, RMS Z)<br>posFileName and posFileType are required to be correctly set for LC (loosely-coupled) Integration |
| **GPS/GNSS position (and velocity) setting for LC (loosely-coupled) integration** | | |
| refRinexObs | '' | reference station rinex observation file name |
| rovRinexObs | '' | rover station rinex observation file name |
| gpsRinexNav | '' | GPS rinex navigation file name |
| gloRinexNav | '' | GLO rinex navigation file name |
| gpsOrbitSp3 | '' | GPS precise orbit file name |
| gloOrbitSp3 | '' | GLO precise orbit file name |
| **GPS/GNSS raw data (reference and rover RINEX observations, broadcast and precise ephermis) for TC (Tightly-coupled) integration** | | |
| sensorType | '' | IMU sensor type (HG1700, H764G, LN100, LN200, MEMS IMU400CC, etc.) |
| imuFileName | '' | IMU binary raw data file name (format 1+6 : time, fxyz, wxyz) |
| navFileName | '' | IMU binary navigation solution file (format 1+9 : time, lat, lon, ht, Vn, Ve, Vd, Heading (yaw), Pitch, Roll) |

| | | |
|---|---|---|
| C | eye(3) | IMU axis rotating matrix to get initial roll and pitch close to 0, this is mainly due to the IMU hardware assembly |
| **IMU (Inertial sensor) settings** | | |
| levelArmOffset | [0.0 0.0 0.0] | GPS Level Arm Offset w.r.t IMU body center, row vector (1x3) |
| **GNSS and IMU spatial relationship settings** | | |
| sigma_pos | 10.0 | initial sigma for position state vector, if the initial position is known, then decrease this value |
| sigma_vel | 0.01 | initial sigma for velocity state vector, normally assumed to be stationary at the begin |
| sigma_roll | 0.0 | initial sigma for roll, if 0.0, the actual value will be set according to the sensorType |
| sigma_pitch | 0.0 | initial sigma for pitch, if 0.0, the actual value will be set according to the sensorType |
| sigma_yaw | 0.0 | initial sigma for yaw, if 0.0, the actual value will be set according to the sensorType |
| imu_sigma_acc_bias | 0.0 | initial sigma for acceleration bias, if 0.0, the actual value will be set according to the sensorType |
| imu_sigma_gyro_bias | 0.0 | initial sigma for gyro bias, if 0.0, the actual value will be set according to the sensorType |
| imu_sigma_acc_scale_factor | 0.0 | initial sigma for acceleration scale factor, if 0.0, the actual value will be set according to the sensorType |
| imu_sigma_gyro_scale_factor | 0.0 | initial sigma for acceleration scale factor, if 0.0, the actual value will be set according to the sensorType |
| sigma_levelArmOffset | 0.0 | GPS Level Arm Offset accuracy (to turn off estimation, set a value <=0.0) |
| **GNSS/IMU integration Kalman filter initial covariance settings** | | |
| noise_acc | 0.0 | accelerometer measurement noise, if 0.0, the actual value will be set according to the sensorType |
| noise_gyro | 0.0 | gyro measurement noise, if 0.0, the actual value will be set according to the sensorType |
| **GNSS/IMU integration Kalman filter driving noise characteristics settings** | | |

| | | |
|---|---|---|
| imaFileName | '' | Image file name |
| **Image sensor settings** | | |
| sigmaNHC | 0.0 | Non-Holonomic Constraint (NHC) sigma (<=0.0 => turn off NHC) |
| sigmaZUPT | 0.0 | Sigma for ZUPT measurement constraints (<=0.0 => turn off ZUPT) |
| autoZUPT_bias_fxyz | 9.8 | bias/sigma/bandwidth for automatically-detect ZUPT using acceleration measurements |
| autoZUPT_sigma_fxyz | 0.05 | bias/sigma/bandwidth for automatically-detect ZUPT using acceleration measurements |
| autoZUPT_bw_fxyz | 20 | bias/sigma/bandwidth for automatically-detect ZUPT using acceleration measurements |
| autoZUPT_bias_wxyz | 0.0 | bias/sigma/bandwidth for automatically-detect ZUPT using gyro measurements |
| autoZUPT_sigma_wxyz | 0.001 | bias/sigma/bandwidth for automatically-detect ZUPT using gyro measurements |
| autoZUPT_bw_wxyz | 20 | bias/sigma/bandwidth for automatically-detect ZUPT using gyro measurements |
| waveletDN_level | 0 | Wavelet-based de-noising (0 => turn off smoothing), required Wavelet toolbox |
| initAlign_second | 5 | seconds used for initial alignment |
| **System parameters settings** | | |
| isOutNED | False | output solution is NED (local coordinate system, North-East-Down) w.r.t a reference starting point or not<br><br>true : => NED (output local coordinate system)<br><br>false: => Lat, Lon, Ht |

| | | |
|---|---|---|
| solOutputFileName | 'KF_sol.txt' | File name for output Kalman filter solutions. File format is: Time (1: GPS week second), NED (2,3,4: m ) or BLH (Lat(2: radian), Lon(3: radian),Ht(4: m)), Vned(5,6,7: m/s), RPY (Roll(8: radian), Pitch(9: radian ), Yaw/Heading(10: radian)), RMS ECEF XYZ (11,12,13: m), RMS Vned(14,15,16: m/s),RMS RPY(17,18,19: radian) |
| errOutputFileName | 'KF_err.txt' | File name for output Kalman filter error state vector. File format is: Time (1: GPS week second), Accelerometer biases (2,3,4: $m/s^2$), gyro biases (5,6,7:radian/s) , accelerometer scale factor errors (8,9,10:unitless), gyro scale factor errors (11,12,13: unitless), antenna level arm offset (14,15,16: m), RMS of accelerometer biases (17,18,19:$m/s^2$), RMS of gyro biases (20,21,22: radian/s), RMS of accelerometer scale factor errors (23,24,25: unitless), RMS of gyro scale factor errors (26,27,28: unitless), RMS of level arm offset (29,30,31: m) |
| ambOutputFileName | 'KF_amb.txt' | File name for output Kalman filter ambiguity fixing information. File format: Time(1), baseline component NED (2,3,4), reference variance (5), LAMBDA ratio (6), number of DD (double-difference) pairs (7), reference satellite ID1 (8), rover satellite ID1 (9), L1 ambiguity (10:cycle), L2 ambiguity(11:cycle), reference satellite ID2 (12), rover satellite ID2 (13), L1 ambiguity (14:cycle), L2 ambiguity(15:cycle), ... |
| outToScreen | true | Print Kalman filter solution (Time, NED/BLH, Vned, RPY) to screen or not |

**Table 5.1** System parameters, input and output settings

### 5.1.3 Sensor Fusion

The sensor integration is done by the function "`gps_imu_filter.m`". This function takes the output from the function "`make_filter_option.m`" and fuses all the sensors' measurements based on the settings. The sensor integration algorithm executes in the following steps:

1. Load TC (Tightly-coupled) integration related data (reference and rover station RINEX data, broadcast and precise GNSS orbit data)

2. Load LC (Loosely-coupled) integration related data (position and velocity data together with associated RMS data)
3. Load IMU data, including the following optional pre-processing: IMU data re-sampling, rotating according to pre-defined rotating matrix, and IMU data wavelet-based de-noising
4. Load initial navigation solution or IMU initial alignment
5. EKF (Extended Kalman filter) navigation filter setup (initial covariance and platform motion noise covariance settings according to pre-defined values or IMU sensor type)
6. Optional autonomous ZUPT detection before filtering
7. Organize related output information
8. Enter inertial sensor measurement loop
    1. Time update of EKF
        - State propagation in time: free inertial navigation using the current inertial measurements (accelerations and angular rates), and previous epoch's navigation solutions
        - Covariance propagation in time:
            - psi-angle inertial linearized error model
            - accelerometer and gyro platform motion noise
    2. Form GNSS single-difference measurements between two receivers if GNSS observations are available at current epoch
        - satellite orbit evaluation
        - add corresponding state vectors (ionospheric delay residuals, L1 ambiguity and L2 ambiguity) for new satellites
        - remove corresponding state vectors for dropped satellites
        - single-differential (SD) code measurement update for all satellites
        - single-differential (SD) carrier-phase measurement update for all satellites
        - Integer AR (ambiguity resolution) search using LAMBDA for L1 and L2

- constraint integer double-difference (DD) ambiguities if DD ambiguities are fixed

3. Position measurement update if position is available
4. Velocity measurement update if velocity is available
5. ZUPT measurement update if stationary condition is detected
6. NHC (Non-holonomic constraint) if NHC option is set
7. Image measurement update if image solution is available
8. Output solution to files and screen
9. Move to next epoch

9. Close output files

The solution visualization is supported by two functions "`plotkf.m`" and "`ploterr.m`". Before calling these two functions, use `load` functions to load Kalman filter solution file to `kf` and error state vector file to `err`. See Appendix 9.1 and 9.2

## 5.2 DEEP INTEGRATION NAVIGATION SOFTWARE

### 5.2.1 Introduction

The deep integration software package was implemented and tested in Matlab programming environment. Main components of the software package include software GPS receiver; strapdown inertial navigation mechanization; inertial aiding of GPS signal accumulation over an extended integration interval for processing of weak signals; and, estimation of inertial drift terms via a tightly coupled Kalman filter. The Matlab development environment was primarily chosen for the convenience of its debugging and graphical representation tools. In addition, Matlab execution performance has been increased recently and is comparable to implementations in C and C++. The main computational load of the deep integration software is associated with the software receiver implementation and, specifically, with its signal correlation engine. In order to accelerate the software into real-time, this engine has to be implemented in a dedicated hardware such as FPGA or a DSP board. The rest of the software functionality is close to real-time performance requirements (even when implemented in Matlab) and does not need any specialized implementation platforms.

This section shortly describes the software design and usage of the GPS/INS deeply integrated software module in the Matlab environment. The design and development of this Matlab-based software is to be served as the SDK (Software Development Kit) for further prototype development of deeply integrated systems.

The major features of the deep GPS/INS integration software module include:

- Software GPS receiver component including generation of replica signals; correlation of replica signals with incoming GPS signals that are down-sampled to a baseband; correlation of incoming and replica signals over a specified interval; wipe-off of navigation data bits; and, estimation of code and carrier phase signal measurements based on correlation results;
- Strapdown INS mechanization including INS initialization; attitude computations and velocity and position integration routines;
- GPS/INS integration using a complementary Extended Kalman filter (EKF) that applies GPS pseudoranges and temporal changes in GPS carrier phase as measurement observables;
- Aiding of GPS signal accumulation using inertial data and satellite ephemeris, which includes computation of changes in code and carrier signal parameters over the accumulation interval;
- 21-dimensional state vector to model the INS position error (3), INS position change error (3), INS velocity error (3), INS attitude/orientation error (3), INS accelerometer bias (3), INS gyro bias (3), GPS receiver clock error states (3);
- Using psi-angle linearized inertial error model;
- Using 6-state driving process noise model (3 accelerometer measurement noise and 3 gyro measurement noise).

The following section provides a basic description of the deep integration Matlab implementation; the software receiver and Kalman filtering details are omitted, and only the input/output and integration functionality are discussed.

### 5.2.2 Sensor Fusion

The sensor integration is done by the function "`Deep_GPS_Inertial_ver1.m`". This function loads test data from specified data files and then implements deep integration processing routines. The integration algorithm executes in the following steps:

10. Load pseudorange range and carrier phase measurements of a stand-alone GPS receiver from a specified file (`RANGE_Aug31.mat` for current settings);

11. Load position solution measurements of a stand-alone GPS receiver from a specified file (`NovAtel_BESTPOS_Aug31`.mat for current settings);

12. Load IMU data from a specified file (`IMU_Data_Aug31_2.mat` for current settings);

13. Load GPS/INS tightly coupled solution from a specified file (`GPSINS_Aug31.mat` for current settings) for comparison purposes;

14. Load software receiver initialization results and ephemeris data (from `Ephemeris_Aug31.mat` and `RCVR_init_Aug31.mat` for current settings);

15. Perform initialization of main structures including:

    - GPS initialization (`GPSStructInitialization3.m`);
    - Preliminary INS initialization (`INSStructInitialization5_2.m` and `INSInitialization_temp.m`);
    - Initialization of software receiver tracking channels (`TrackingChannels_Init.m`);
    - Initialization of Kalman filter (`FilterStuctInitialization_temp.m`);

16. Enter deep integration processing loop

17. Read IMU measurements from a data file (`GetIMUMeasurements.m`)

18. Perform strapdown INS updates (`INSNavUpdates4.m`);

19. Perform Kalman filter prediction updates (`KalmanFilterPredictions.m`);

20. If the INS navigation solution is initialized, compute dynamic reference trajectory for extended accumulation of GPS signals (`AidingTrajectory.m`);

21. Read measurements of the unaided GPS receiver (`GetGPSMeasurements3_5.m`);

22. Compute GPS position and velocity solution and, if the INS navigation solution is initialized, compute Kalman filter observables for the unaided GPS receiver (`KalmanFilterGPSObservables7_5.m`);

23. Save position and velocity solution of the unaided receiver into an output array structure;

24. If the INS navigation solution is initialized, perform deep integration acquisition of weak GPS signals (`DeepIntegration_Acquisition.m`);

25. If the INS navigation solution is initialized, perform deep integration tracking of weak GPS signals (`DeepIntegraton_Tracking.m`);

26. If the INS navigation solution is initialized, compute Kalman filter observables of the deeply integrated receiver (`DeepIntegration_KalmanFilterGPSObservables.m`);

27. If the INS navigation solution is initialized, save GPS/INS residuals of delta position and position into output data arrays;

28. If the INS navigation solution is initialized, perform Kalman filter estimation updates (`StateEstimation2.m`);

29. If the INS navigation solution is not initialized, attempt to perform "in-flight" INS initialization based on GPS position and velocity measurements (`INSInitialization6.m`);

30. If the INS initialization process is just being completed, update the Kalman filter structure accordingly (`FilterInitialization1_2.m`);

31. Reset INS error states (`AgentStateReset7.m`);

32. Save data into output arrays including deep integration position estimates (ENU and LLH format); velocity estimates; gyro drift and accelerometer bias estimated; and, Kalman filter covariances;

33. Exit the loop;

34. Plot processing results.

# 6. LASER SENSOR DEVELOPMENTS

## 6.1 Introduction

Laserscanning systems have seen enormous growth in the geospatial data acquisition industry in the past decade, mainly because of the excellent performance offered by this technology and the explicitness of the 3D data; a review on LiDAR technology can be found in (Shen and Toth, 2008) and on advanced data processing in (Vosselman and Maas, 2010). More importantly, laser sensing technologies have seen remarkable developments in the past few years. The single laser sensor-based systems still dominate the geospatial market but new multi-sensor and integrated array sensors have been introduced and increasingly used recently. At the beginning of the project, only single sensor-based systems were available where mechanical scanning provides for either profile or 2D range measurements. While these solutions provide good measurement performance, the slowness of the scanning has two drawbacks: (1) artifacts are introduced in dynamic environment, and (2) the data acquisition is time-consuming. Multi-sensor systems provide significant performance improvement, as a linear array of is used for scanning, and thus, a profile can be directly measured in a short time or one dimension is eliminated from 2D scanning. Consequently, these systems typically offer two orders of improvement in speed, and clearly, they are fast enough for considering their use in a reengineered version of the prototype used for testing in this project. In the integrated array sensor category, Flash LiDAR systems provide excellent performance, acquiring 2D depth images at 10-30 image/second rates. In fact, this fast observation capability is not needed to meet the geolocation requirements of this project.

Flash LiDAR, also called Flash LADAR, is a substantially different sensing technology compared to pulsed and CW LiDAR techniques, as it is based on an sensor array, so a it can capture a whole 3D, also called depth or range, image with intensity data in a single step. Flash LiDAR can use both basic solutions to emit laser, either a single pulse with large aperture will “flash” the area for a short time or in CW mode a continuous laser “light” provides steady illumination in the area. One of the first and early Flash LiDAR models, the SWR3000 (Kahlmann, 2006) is based on CW approach, offering an operating range up to 7.5 m and a frame rate of 15 Hz. All the Flash LiDAR systems are based on solid state semiconductor technologies, and there is a large variety of solutions and, consequently, operating parameters. Advanced modern systems can reach the 1,500 m range, which makes them deployable on airborne vehicles. Because of their smaller size, and less power requirements, they are attractive for mobile platforms. Flash LiDAR technology goes back to the late 90’s, but started to reach maturity just recently. The issues are the limited power that should illuminate the area and the complex circuitry, typically avalanche photodiode detector (APD), needed to detect the few photons, backscattered from objects. There are several Flash LiDAR systems in low-end category, while professional-grade systems are mainly manufactured by two companies, Advanced Scientific Concept, Inc., offering three products (ASC), and Raytheon Vision

Systems, mainly focused on sensor developments (Bailey *et al.*, 2010). Flash LiDAR systems have been rapidly advancing in both the professional and commercial markets; the typical sensor array size is in the 128x128 and 256x256 range. Airborne Flash LiDAR systems are already available in military reconnaissance and are increasingly tested in civilian mapping too. In the commercial market, there are many small and inexpensive systems, which may only work indoors, as the emitted laser energy of these sensors is very low and ambient radiation outdoors prevent reliable operations.

To assess the feasibility of Flash LiDAR for the purpose of this project, experiments were conducted using the Microsoft Kinect sensor. Clearly, this sensor is not robust to be considered for a deployable geolocation system, yet it has all the characteristics that can expected from future professional grade Flash LiDAR systems. Note that strictly speaking, Kinect is not a Flash LiDAR sensor, as it uses structured near IR light for depth recovery, but for practical reasons, it can be viewed as Flash sensor, as it provides identical depth images.

## 6.2 Kinect Sensor

The Kinect sensor is a motion sensing input device for the Xbox 360 video game console, originally developed by PrimeSense (PrimeSense), and acquired by Microsoft. The primary purpose is to enable users to control and interact with the Xbox 360 through a natural user interface using gestures and spoken commands without the need to touch a game controller at all. The Kinect has three primary sensors: a Flash LiDAR (3D camera), a conventional optical RGB sensor (2D camera), and microphone array input. The device is USB-interfaced, similar to a webcam, and appears as a "black box" for the users.

Very little is known of the sensors, internal components and processing methods stored in the firmware. The laser, IR, emitter projects a structured light pattern of random points to support 3D recovery. The 2D camera can acquire standard VGA, 640x480, and SXGA, 1280x1024, images at 30 Hz. The color formation is based on Bayer filter solution, transmitted in 32-bit and formatted in the sRGB color space. The FOV of the 2D camera is 57° x 43°. The 3D camera can work in two resolutions with frame sizes of 640x480 and 320x240. The range data comes in 12-bit resolution. The sensors' spatial relationship is shown in Figure 6.1. The approximate difference between the laser emitter and detector that form a stereo par is about 7.96 cm, and the baseline between the 2D and 3D cameras is about 2.5 cm.

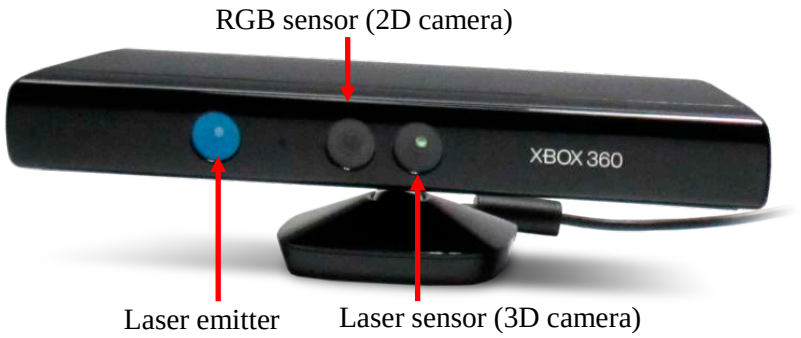



**Figure 6.1** Kinect XBOX 3600 sensor, including 2D and 3D imaging sensors

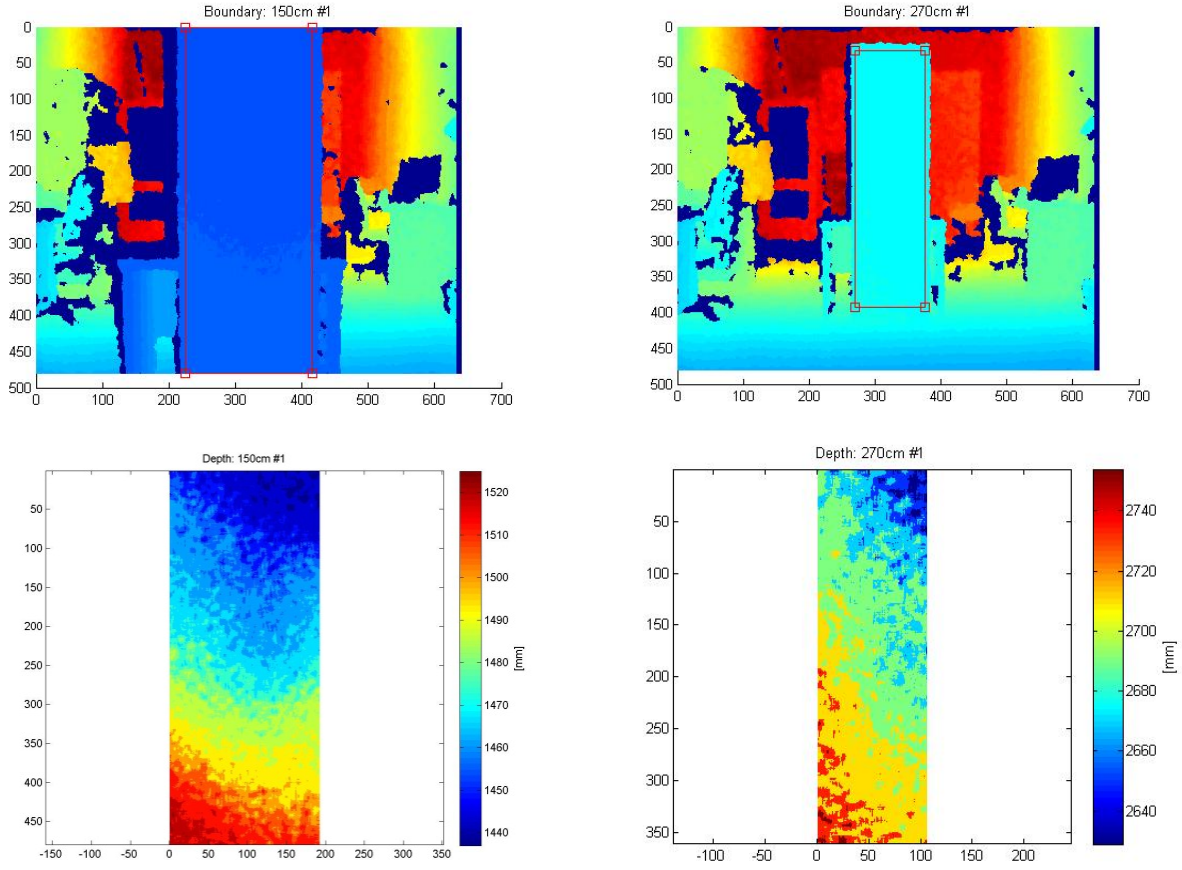
Microsoft provides an SDK (Windows 7, Visual Studio 2010 Enterprise, and DirectX) to support application developments, including both polled and event-based access to the image data streams (Microsoft). In addition, skeletal tracking of up to two people is also supported. Kinect has a default measuring range of 0.8 m and to 3.9m (no ambiguity), which can be extended; our experiences indicate that up to 10 m range, reliable depth images can be acquired. The available open source drivers provide additional the opportunity to acquire raw data and a very powerful SDK is also available. In our investigation the SensorKinect driver (Github) was used with OpenNI (OpenNI) and all the subsequent processing was done in Visual Studio C++ and Matlab.

The Kinect sensor installation in a backpack configuration used in our testing is shown in Figure 6.2


**Figure 6.2** Personal navigator (PN) sensor assembly with Kinect sensor

## 6.3 Sensor Repeatability Test

The repeatability of the range measurement is an essential aspect of using depth imaging sensors, as it provides the assessment of the ranging precision in short term. To determine the sensor repeatability performance, a planar target was imaged from a distance ranging from 0.5 m to 5 m in 0.1 m steps. Figure 6.3 shows 3D (depth) images of the target from two different distances.

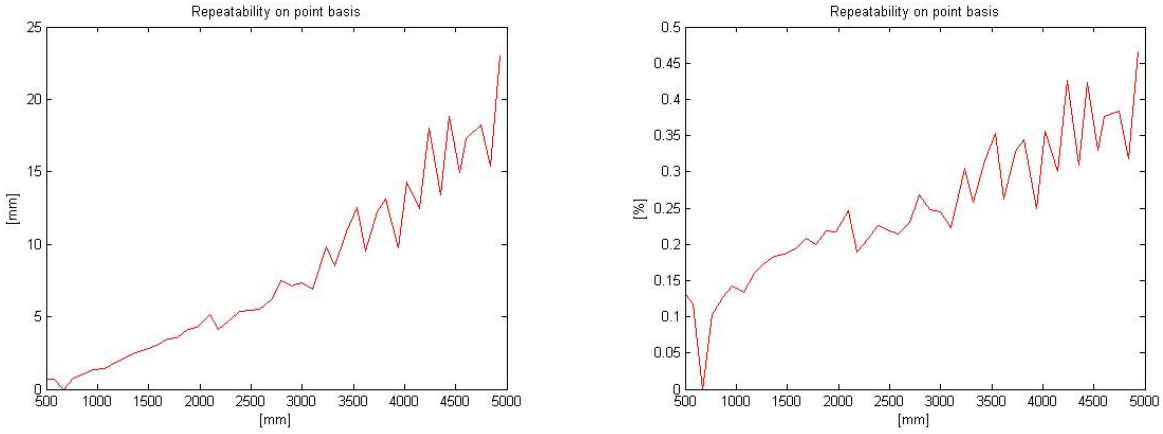



(a) (b)

**Figure 6.3** Pseudo color 3D images taken at 150 (a) and 270 (b) cm ranges; first row entire images and second raw images of the planar target extracted

The measurement was repeated six times, so a total of 46 x 6 images were acquired and processed. The planar target has a size of 180 cm x 60 cm, so its FOV in the image changes a lot. Consequently, the number of points obtained by the 3D sensor from the reference planar target varies over a large range, from 200K down to 10K.

In the first step, the standard deviation was computed on a point basis for each distance. The repeatability results, shown in Figure 6.4, clearly indicate a near linear dependency on the range; note a quadratic function is the theoretical model. The overall performance for the whole range is lower than 0.5%, which is quite excellent compared to earlier Flash LiDAR results (Kahlmann *et al.*, 2006).




(a) (b)

**Figure 6.4** Repeatability results; (a) absolute and (b) relative performance

## 6.4 Plane Fitting Performance

To achieve a better performance characterization, plane fitting was performed based on principal point component analysis and the fitting error was calculated in the plane normal direction. The fitting error, shown in Figure 6.5, has an interesting shape, as the curve has a local maximum in the central part of the range, near the ambiguity range. There is no obvious explanation for this character, except that the decreasing number of points can result in improving fits. In addition, this curve, as well as all error curves, looks sort of jagged or "discontinuous" which could be partially associated with rounding up the numbers during internal processing of the sensor firmware.

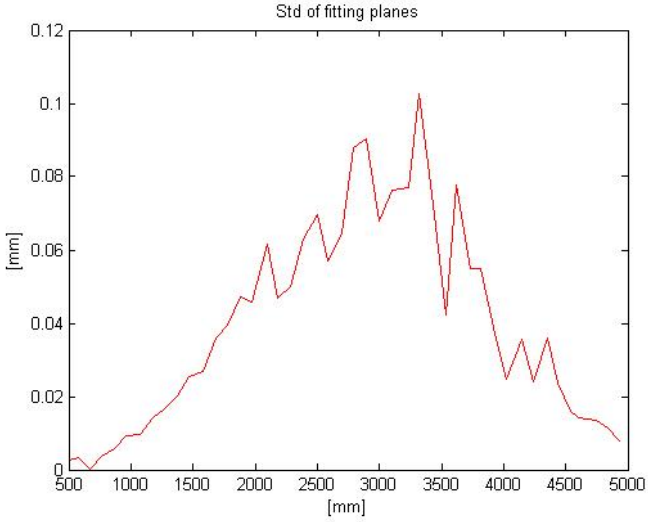
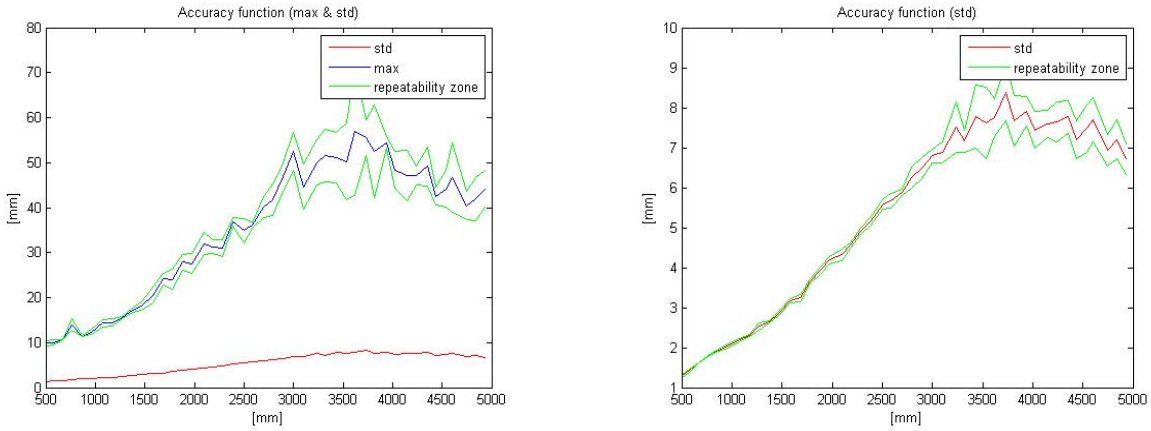



**Figure 6.5** Standard deviation of plane fitting error in surface normal direction

Based on the six measurement sets, the fitting plane residual errors were calculated and basic statistical parameters were determined, including maximums and STD, for each range. Figure 6.6 shows the results, including a maximum error envelope and the STD (a) as well as well as only the STD with error envelope (b). The results clearly indicate good accuracy performance, as at the shortest object distance, the STD is lower than 1 mm and the maximum error is 1 cm, while at 3.5 m (the ambiguity limit) the STD is 7 mm and the maximum is 5 cm. Theoretically, the STD function should be of quadratic form based on the used calculation method, yet the curve looks almost linear. Normalized for the range, the STD is about 0.2% of distance while the maximum error is about 1.6%, as shown in Figure 6.7.




(a) (b)

**Figure 6.6** STD of residual surface fitting errors

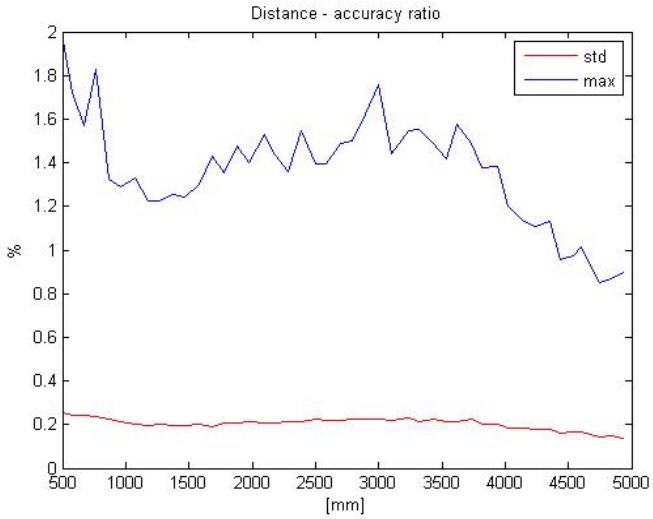
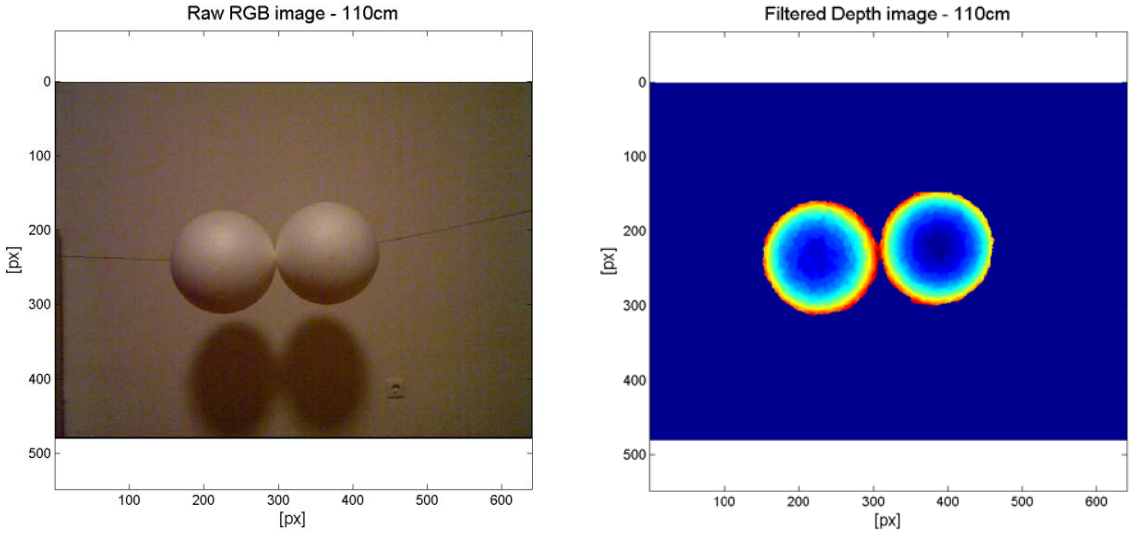



**Figure 6.7** Normalized statistical parameters

## 6.5 Sphere Fitting Performance

The sphere fitting performance is of high importance for the project, as it has a primary impact on the relative geolocation performance. Note it, obviously, comes after repeatability, but the repeatability tests performed for the spheres resulted in similar performance compared to the planar surface based tests. Therefore, to assess sphere fitting performance, several tests were carried out using highly accurate wood balls and then the very same basketballs used in the field tests in Phase I of the project. Figure 6.8 shows optical and depth images of the precise balls used for initial testing.




(a) (b)

**Figure 6.8** Images of precise wood targets

The captured point clouds were filtered and then two spheres were fitted; the radius, center points and fitting error were analyzed (Molnar *et al.*, 2012). Figure 6.9 shows example fitting residuals for one of the spheres at a distance of 100 cm. The results show that there is no significant error around the edges of the spheres indicating that there is a low correlation with respect to the angle of incidence.

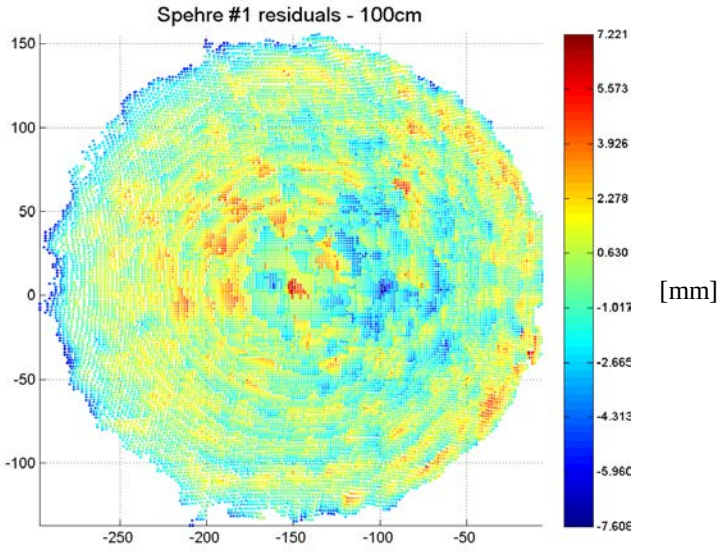



**Figure 6.9** Sphere fitting residuals at sphere to Kinect distance of 100 cm

The analysis of the fitting w.r.t object distance produced unexpected results, as over 2 meters, the value of the radius was getting lower, as shown in Figure 6.10. The repeated measurements, even with multiple Kinect devices, provided the same results. The decreasing number of points with respect to target range cannot be the reason of this error because each sphere is covered by about 5K points at 2 meter distance. To identify the source of this error, a new test was performed, where in the new arrangement two hemispheres were placed on a flat surface, giving the opportunity to find errors caused by peripheral points with bad conditions, see Figure 11. The gap between the plane and spheres is virtually zero, which means that most of the emitted points return even at bad incident angle. In this arrangement, it is possible to check the distance between the closest points of sphere and the plane and compare to the nominal radius of sphere. Thus, looking at the Kinect raw data, the answer to the downscaling turned out to be quite simple. Since the raw measurement values are transmitted over USB interface with a limited transmission capacity with respect to the large amount of image data, the depth values are transmitted as 11-bit integers. Thus, due to this coarse quantization, the spacing is growing by the measurement distance. For instance, a sphere with 15 cm radius has about 30 depth levels at 90 cm object distance and only four at 3 meters, see Figure 6.12.

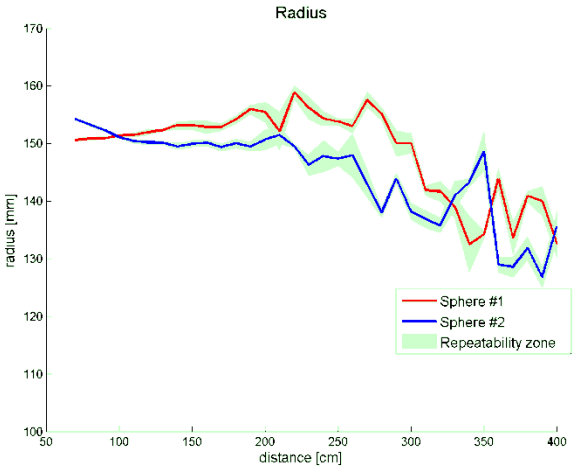



**Figure 6.10** Downscaling effect

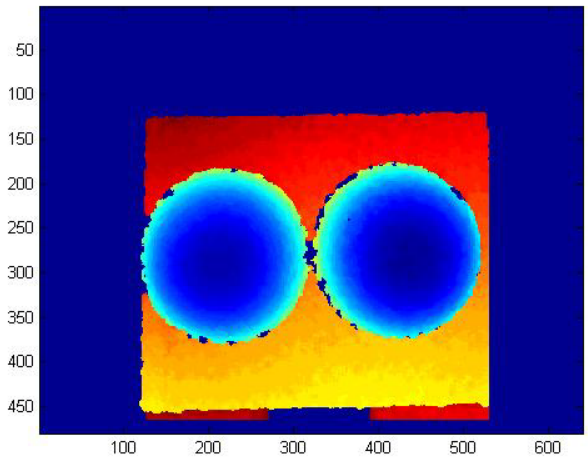
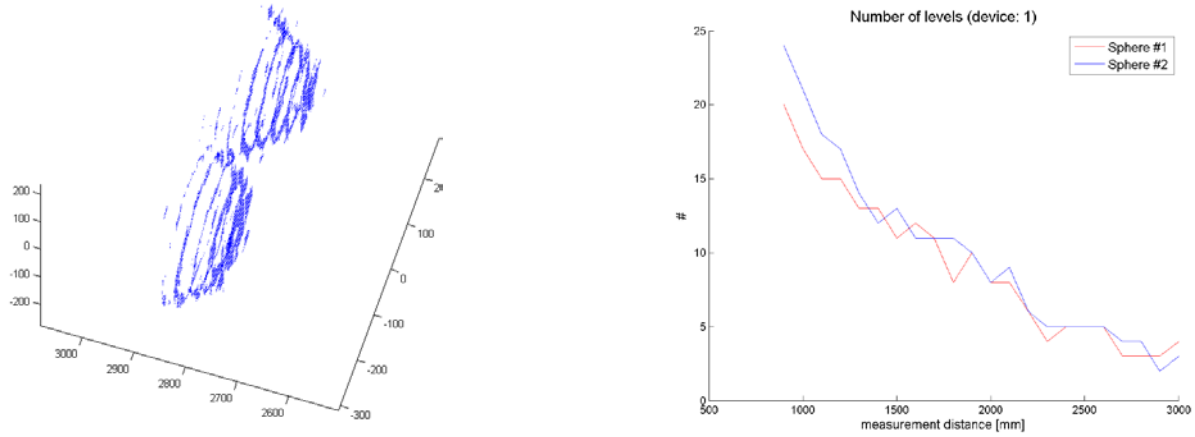



**Figure 6.11** Hemisphere targets on planar surface (depth image)




(a) (b)

**Figure 6.12** Levels on a sphere and level number as a function of distance

The cross-section and the impact of the quantization, “ringing effect” are shown in Figure 6.13.

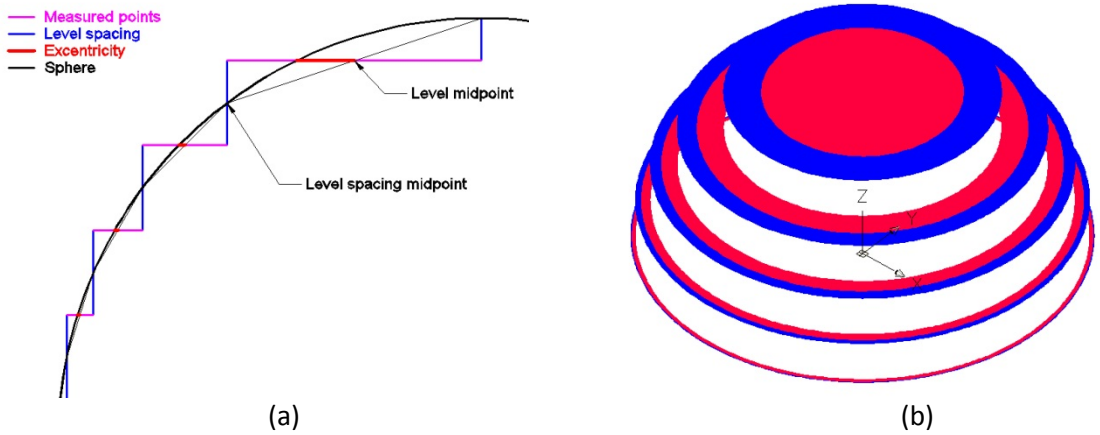



**Figure 6.13** Source of downscaling (a) and quantization error with sign (b).

Knowing the nature of the quantization introduced error, the sphere fitting algorithm can be modified to account for this limitation (Molnar *et al.*, 2012). Then, the accuracy test was repeated with two hemispheres in the backplane arrangement with a distance range of 90-300 cm in 10 cm steps. During the process of fitting spheres with the new fitting method, the radius of spheres, center points' distance and their differences were analyzed. Based on the repeated measurements, the STD of these values was also calculated at each step, indicating a less than 10 mm repeatability accuracy at each distance. The new fitting method performed well and confirms that the raw device measurement itself has good accuracy and only the quantization limits the performance of farther object measurements. By analyzing the radius-distance function, shown in Figure 6.14, it can be concluded that the maximum difference from the directly measured radius (152 mm) is less than 1 cm and STD is less than 3.5mm.

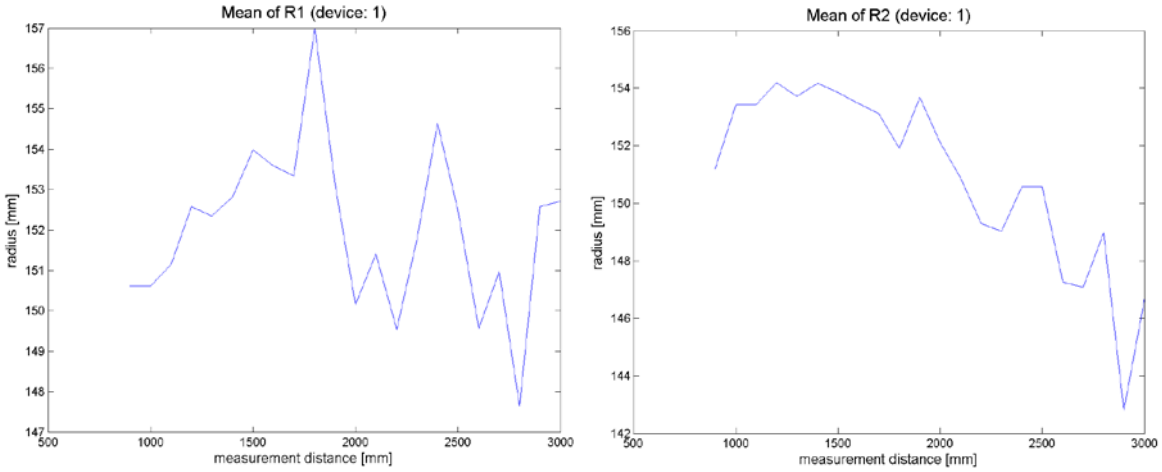



**Figure 6.14** Radius-range function of the two spheres

Compared to previous results, the new tests show that the STD of fitted radiuses has a quadratic form, as depicted in Figure 6.15; note that it was originally expected (Khoshelham, 2011). This proves that the raw depth values are more accurate compared to the quantized ones. In addition, it can be confirmed that the quantization error is significantly higher than measurement error. Thus, if the fitting is performed by the common fitting method, the error is about the 0.5% of object range, while for the modified method, it is less than 0.3% at 3 meter and 0.1% at 1 meter.

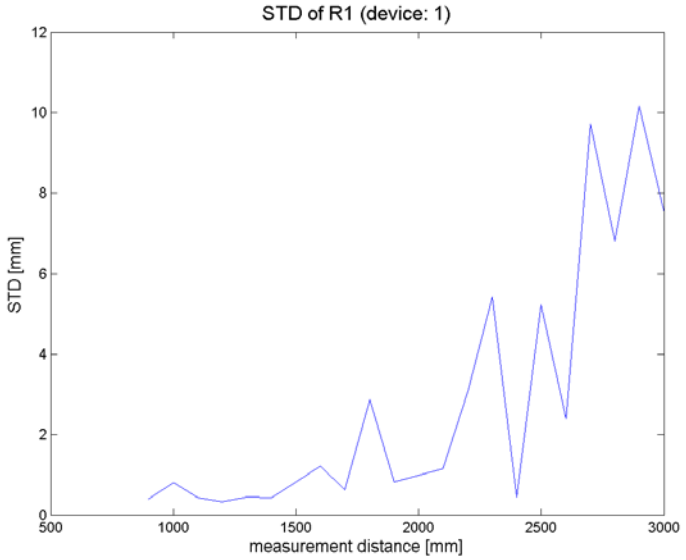
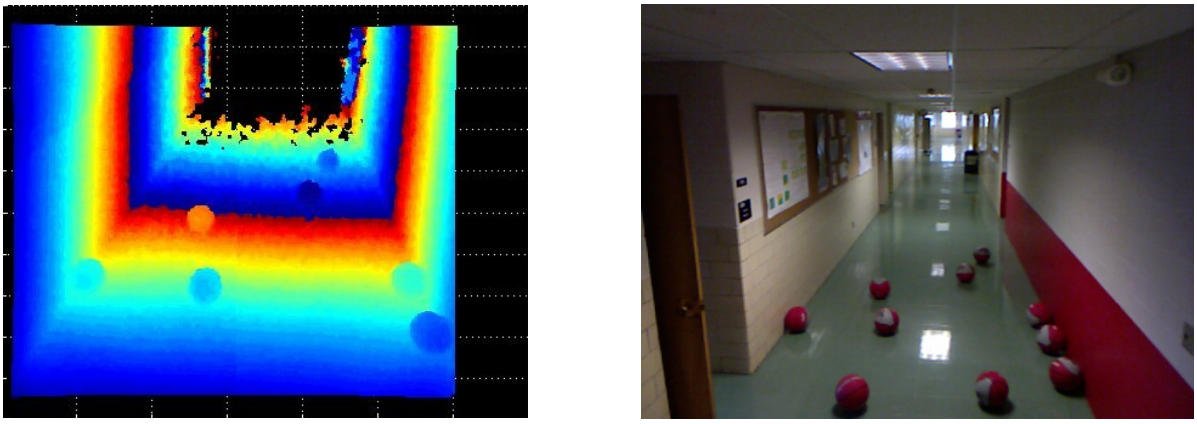



**Figure 6.15** STD of radius has a quadratic form

Finally, tests were carried out with basketballs, and resulted in identical results. A sample image is shown in Figure 6.16; not these targets were used in the field tests of Phase I.



(a) (b)

**Figure 6.16** Images of basketball targets used for performance testing

## 7. CONCLUSION FUTURE RECOMMENDATION

This report discusses the application of the deeply integrated GPS/INS navigation approach for precise positioning in dense canopy environments. A top-level review of the deeply integrated system architecture is offered. Experimental results are presented to demonstrate the deep integration performance for scenarios including suburban tree-covered areas and dense forestry tests. Example results demonstrate that the deep integration supports reliable trajectory reconstruction with the positioning precision estimated to be at a sub-meter level. In order to support the deep integration functionality, the GPS receiver signal processing part needs to be integrated with a navigation-grade or a tactical-grade inertial measurement unit.

Test results presented in the document demonstrate that the use of deeply integrated solution is sufficient for enabling sub-meter level precise geo-referencing capabilities. This level of positioning precision is sufficient to support initial scans of MEC sites that localize anomalous electromagnetic signatures. For a more detailed analysis of a potential MEC location, cm-accurate relative positioning is required, which cannot be supported by deep integration techniques. Hence, it is envisioned, that the incorporation of deep integration into the quadruple system architecture will allow for the removal of the PL system component. However, TLS still have to be included to provide cm-level precise estimates of the relative position.

Recent developments in laserscanning technology, in particular, in Flash LiDAR sensors may provide a basis for efficient implementation of the TLS component of the multisensory system in the near future. The initial tests with simulated sensors confirmed that cm-level accuracy is easily available with randomly deployed spherical targets.

## 8. REFERENCES

Bailey, S., McKeag, W., Wang, J., and Jack, M., 2010. Advances in HgCdTe APDs and LADAR receivers, Infrared Technology and Applications XXXVI, Proc. SPIE 7660, 76603I.

Bhattacharyya, S., Gebre-Egziabher, D., "Development and Validation of Parametric Models for Vector Tracking Loops," NAVIGATION, Journal of the Institute of Navigation, Vol. 57, No. 4, 2010-2011.

Farrell, J. L., "Full Integrity Testing for GPS/INS", NAVIGATION, Journal of the Institute of Navigation, Vol. 53, No. 1, 2006.

Farrell, J. L., "GPS/INS-Streamlined," NAVIGATION, Journal of the Institute of Navigation, Vol. 49, No. 2, 2002.

Github, https://github.com/avin2/SensorKinect, last access in October 2012.

Grejner-Brzezinska, D., Toth C., Sun H., Wang X., "Novel Geolocation Technology for Geophysical Sensors for Detection and Discrimination of Unexploded Ordnance," Proceedings of IEEE/ION PLANS 2008, Monterey, CA, May 2008.

Grejner-Brzezinska, D.A., C. K. Toth, H. Sun, X. Wang, and C. Rizos (2011): A Robust Solution to High-Accuracy Geolocation: Quadruple Integration of GPS, IMU, Pseudolite and Terrestrial Laser Scanning, *IEEE Transactions on Instrumentation and Measurement*, Vol. 60, Num. 11, pp. 3694–3708, 2011.

Grover Brown, R., "A Baseline RAIM Scheme and a Note on the Equivalence of Three RAIM Methods," Global Positioning System: Papers Published in NAVIGATION, Vol. 5, 1998.

Gunawardena, S., Soloviev, A., van Graas, F., "Wideband Transform-Domain GPS Instrumentation Receiver for Signal Quality and Anomalous Event Monitoring," NAVIGATION, Journal of the Institute of Navigation, Vol. 53, No. 4, 2007.

Kahlmann, T., Remondino, F., Ingensand, H., 2006. Calibration for increased accuracy of the range imaging camera SwissRanger, ISPRS Commission V Symposium 'Image Engineering and Vision Metrology', Dresden, Germany, pp. 136-141.

Microsoft, http://www.microsoft.com/en-us/kinectforwindows/resources/, last accessed in October 2012

Molnar, B., Toth, C. K., Detrekoi, A., 2012: Accuracy test of Microsoft Kinect for human morphologic measurements, Int. Arch. Photogrammetry and Remote Sensing, Spatial Inf. Sci., XXXIX-B3, 543-547.

OpenNI, http://openni.org/Documentation/ProgrammerGuide.html, last accessed in October 2012.

PrimeSense, (http://www.primesense.com), last accessed in October 2012.

Shan, J., Toth, C.-K., 2008. Topographic Laser Ranging and Scanning: Principles and Processing. Boca Raton, FL: Taylor & Francis.

Soloviev, A., Bruckner, D., van Graas, F., Marti, L.,"Assessment of GPS Signal Quality in Urban Environments Using Deeply Integrated GPS/IMU," Proceedings of the Institute of Navigation National Technical Meeting, San Diego, Cam January 2007.

Soloviev, A., Gunawardena, S., van Graas, F., "Deeply Integrated GPS/Low-Cost IMU for Low CNR Signal Processing: Concept Description and In-Flight Demonstration," NAVIGATION, Journal of the Institute of Navigation, Vol. 54, No. 1, 2008.

Soloviev, A., Gunawardena, S., van Graas, F., "Decoding Navigation Data Messages from Weak GPS Signals," IEEE Transactions Transactions on Aerospace and Electronic Systems, Vol. 45, No. 2, 2009.

Soloviev, A., C. Toth, and D. Brzezinska (2012): Performance of deeply integrated GPS/INS in dense forestry areas, *Journal of Applied Geodesy*, Vol. **6** (2012), pp. 3–13.

Toth,K. C., Molnar, B., Zaydak, A., Grejner-Brzezinska, A. D, 2012. Calibrating the MS Kinect Sensor, ASPRS 2012 Annual Conference. Sacramento, USA

van Graas, F., Soloviev, A., "Precise Velocity Estimation Using a Stand-Alone GPS Receiver," *Navigation, Journal of the Institute of Navigation*, Vol. 51 No. 4, 2004.

van Graas F., Soloviev A., Uijt de Haag M., Gunawardena S., "Closed Loop Sequential Signal Processing and Open Loop Batch Processing Approaches for GNSS Receiver Design," *IEEE Journal of Selected Topics in Signal Processing*, Vol. 3, Issue 4, 2009.

Vosselman, G., Maas, H.-G., 2010. Airborne and Terrestrial Laser Scanning. Whittles Publishing, Caithness, Scotland, UK.

Wendel, J., Meister, O., Monikes, R., Trommer, G.F., "Time-Differenced Carrier Phase Measurements for Tightly Coupled GPS/INS Integration," Proceedings of IEEE/ION PLANS 2006, San Diego, CA, April 2006, pp. 54-60.

# 9. APPENDIX

## 9.1 AIMS PRO: Major functions

| Function name | Description |
|---|---|
| ang2qua.m | Angle convert to quaternion |
| att2c_nb.m | Attitude convert DCM |
| att2qua.m | Attitude to quaternion |
| b2n.m | Body frame to navigation frame |
| blh2C_en.m | Compute rotating matrix from e-frame to navigation frame |
| blh2xyz.m | Lat, Lon and Ht to ECEF XYZ |
| calgravmodel.m | Compute gravity model |
| calomega_nen.m | Compute $\Omega_{en}^{n}$ |
| calomega_nie.m | Compute $\Omega_{ie}^{n}$ |
| calradius.m | Compute Rn and Rm |
| checkzupt.m | Automatically ZUPT detection |
| coarsealign1.m | Coarse alignment |
| coarsealign2.m | Coarse alignment |
| coarsealign3.m | Coarse alignment |
| coarse_alignment.m | Coarse alignment |
| c_nb2att.m | DCM to attitude |
| c_nb2qua.m | DCM to quaternion |
| geoatan1.m | Arctan |
| geoatan2.m | Arctan |
| geoconst.m | Related constant definition |

| | |
|---|---|
| gps_imu_filter.m | Main Sensor fusion |
| imuwavedenoise.m | Wavelet-based de-noising |
| make_filter_option.m | Make filer option, main system setting function |
| n2b.m | Navigation frame to Body frame |
| n2e.m | Navigation frame to e-frame |
| nav_ffun.m | Free inertial navigation |
| ploterr.m | Plot error state vector |
| plotkf.m | Plot KF estimations |
| process_noise_model.m | KF process noise model |
| psi_angle_error_model.m | KF psi-angle error model |
| qua2att.m | Quaternion to attitude |
| qua2C_nb.m | Quaternion to DCM |
| quamulti.m | Quaternion multiply |
| readdata.m | Read binary data |
| readposdata.m | Read text postion & velocity data |
| readrinexobs.m | Read Rinex 2.10 observation data |
| resampleimudata.m | IMU data re-sampling |
| skew.m | Skew matrix |
| xyz2blh.m | ECEF XYZ to lat, lon, ht |

## 9.2 AIMS PRO: Example

```
options.posFileType = 1;
options.posFileName = '..\20040704\data\gpsamb.bin'; % GPS position time,
lat, lon, ht, RMS X, RMS Y, RMS Z
options.imuFileName = '..\20040704\data\hg764_imu.bin'; % IMU Binary File 1+6
options.navFileName = '..\20040704\data\finenav_hg764.bin'; % NAV Binary File
1+9
options.sensorType = 'H764G';
```

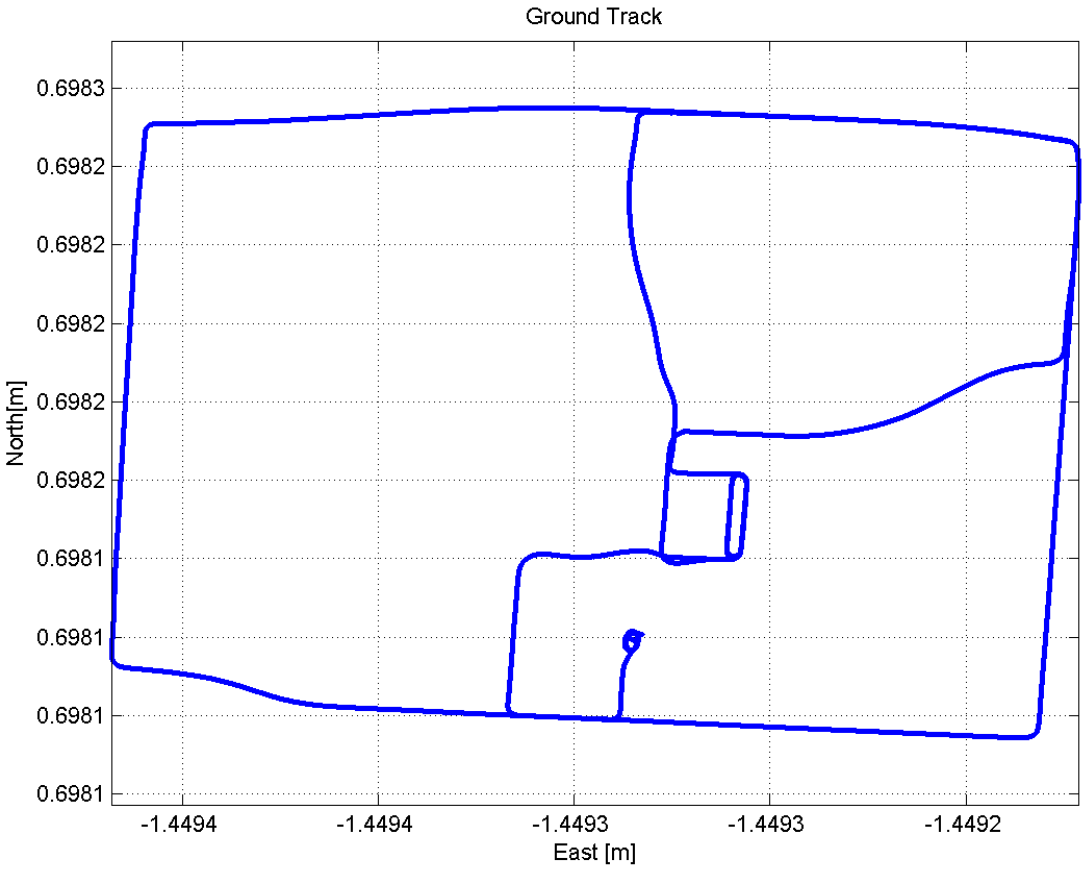



**Figure 9.1** Ground track

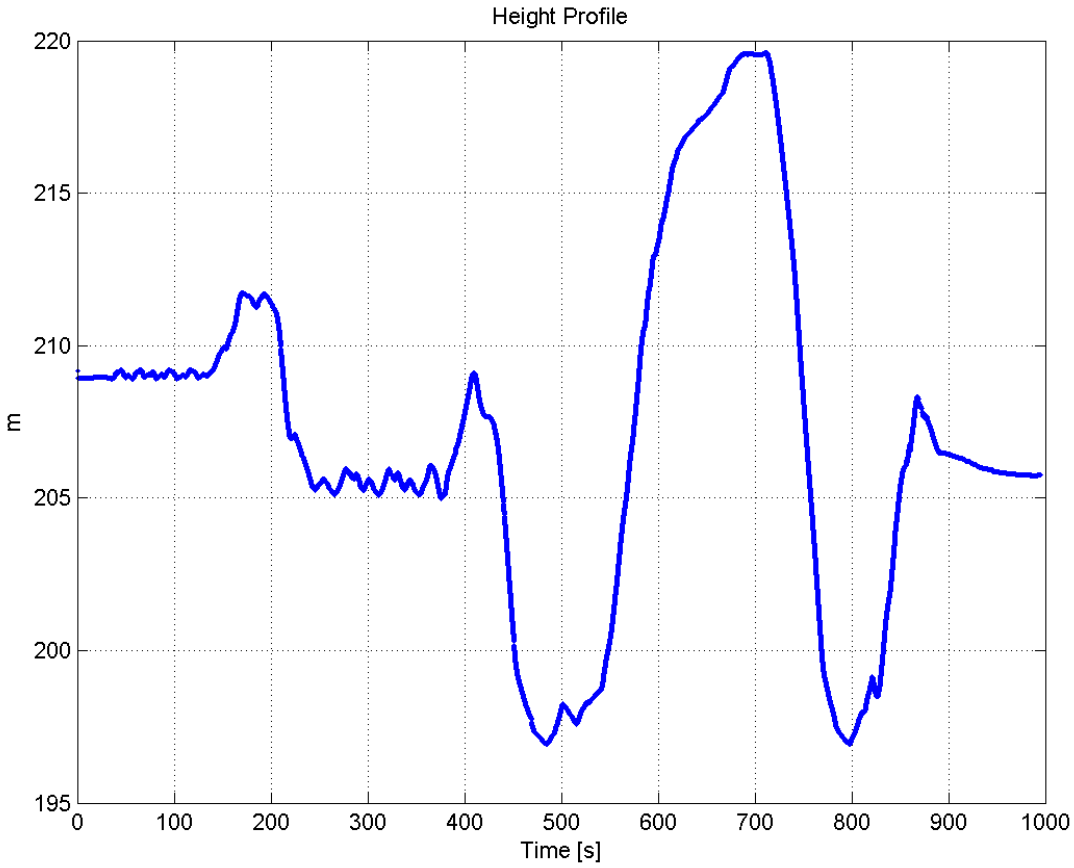



**Figure 9.2** Height profile

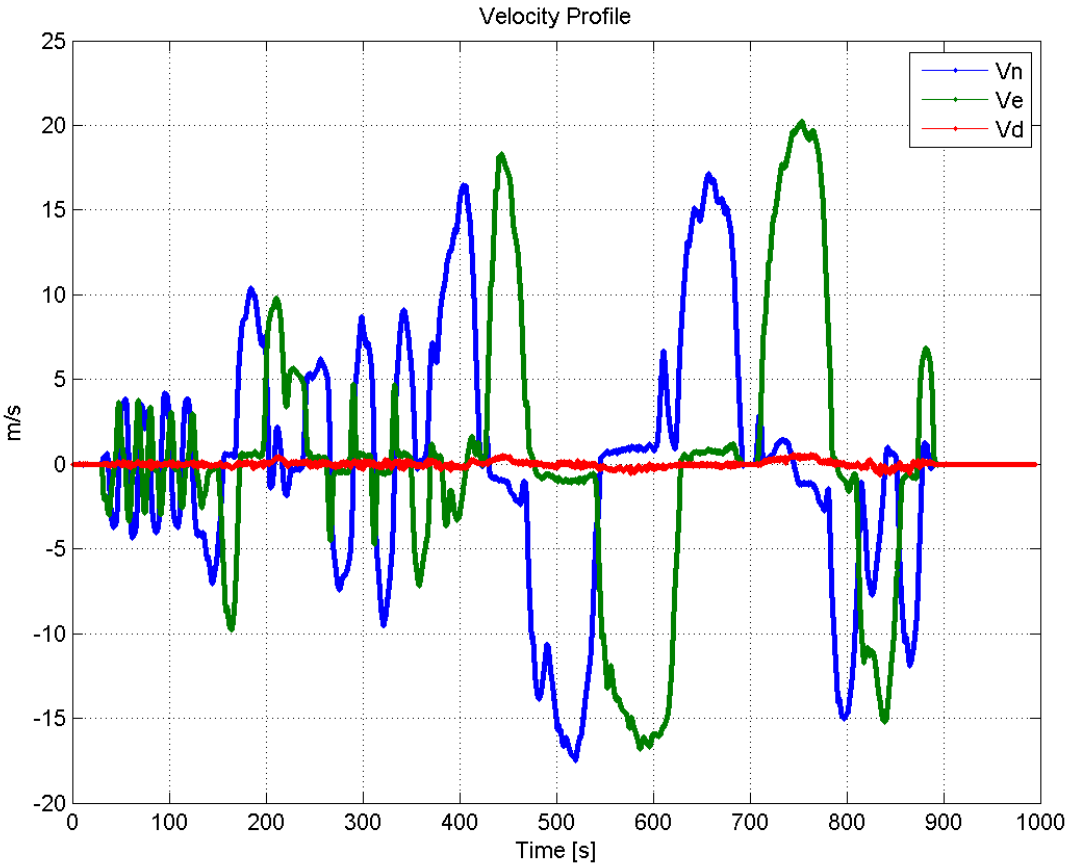



**Figure 9.3** Velocity profile

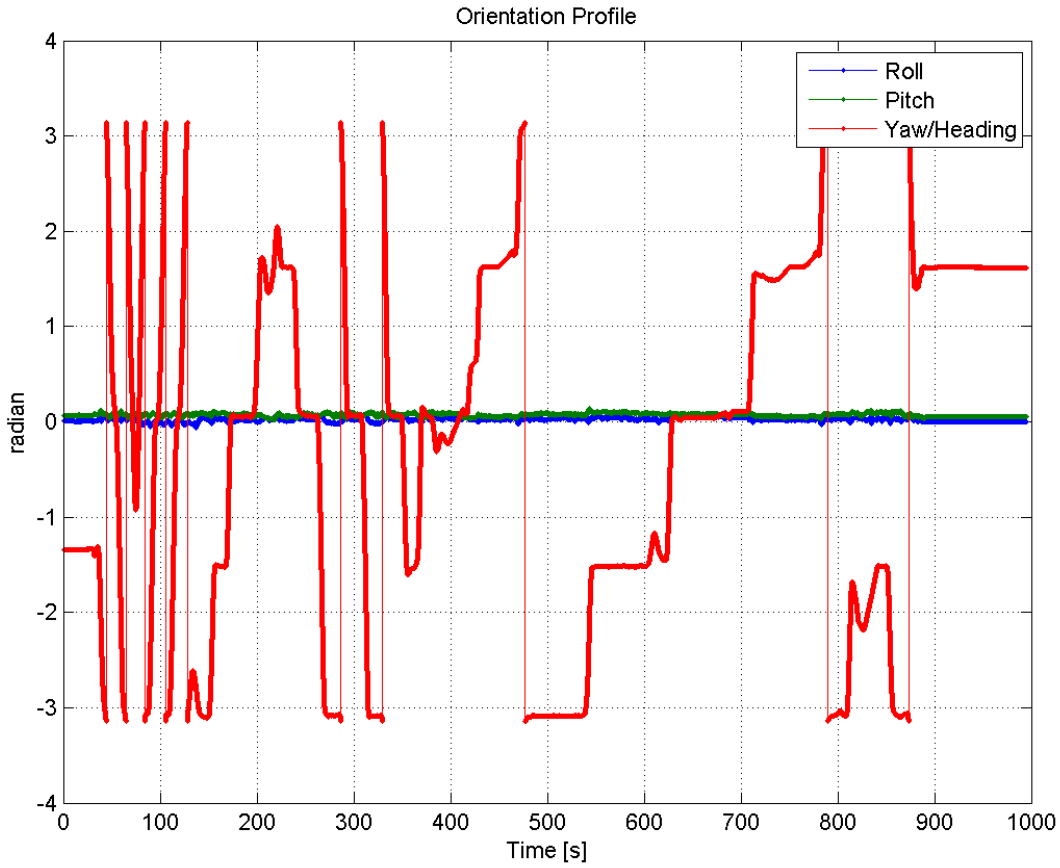



**Figure 9.4** Orientation profile

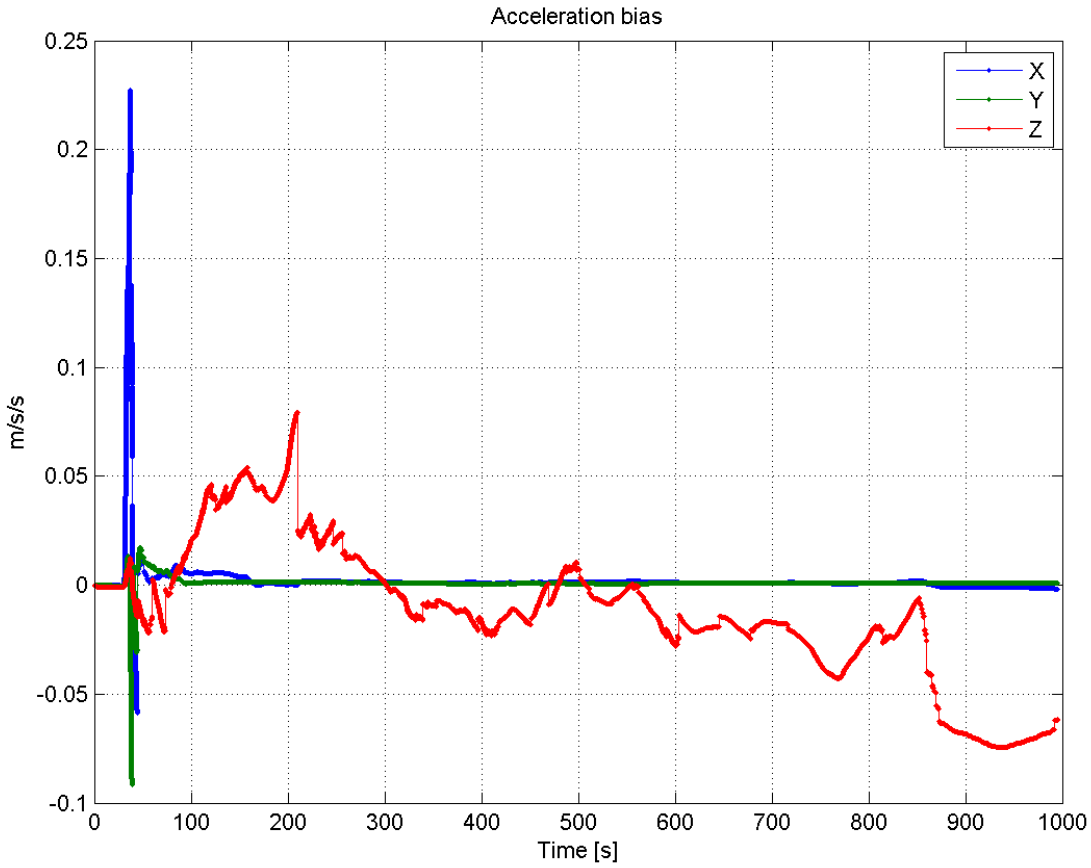



**Figure 9.5** Acceleration bias estimation

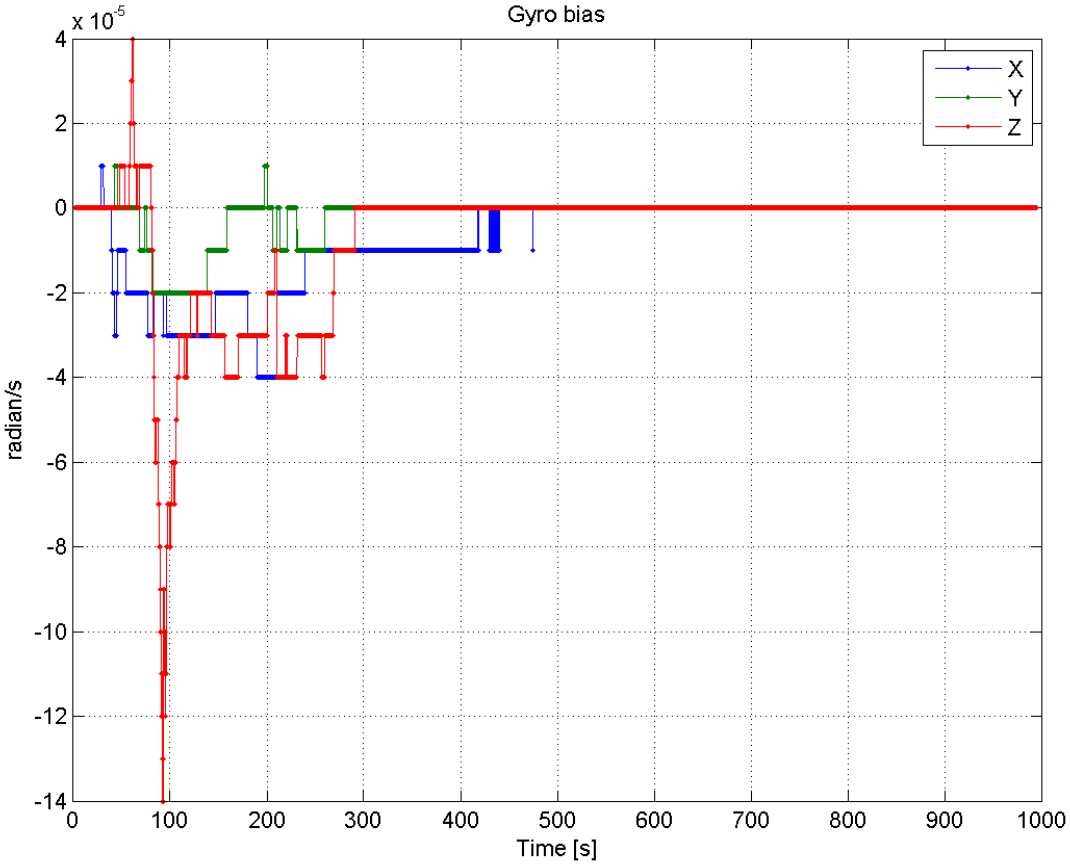



**Figure 9.6** Gyro bias estimation

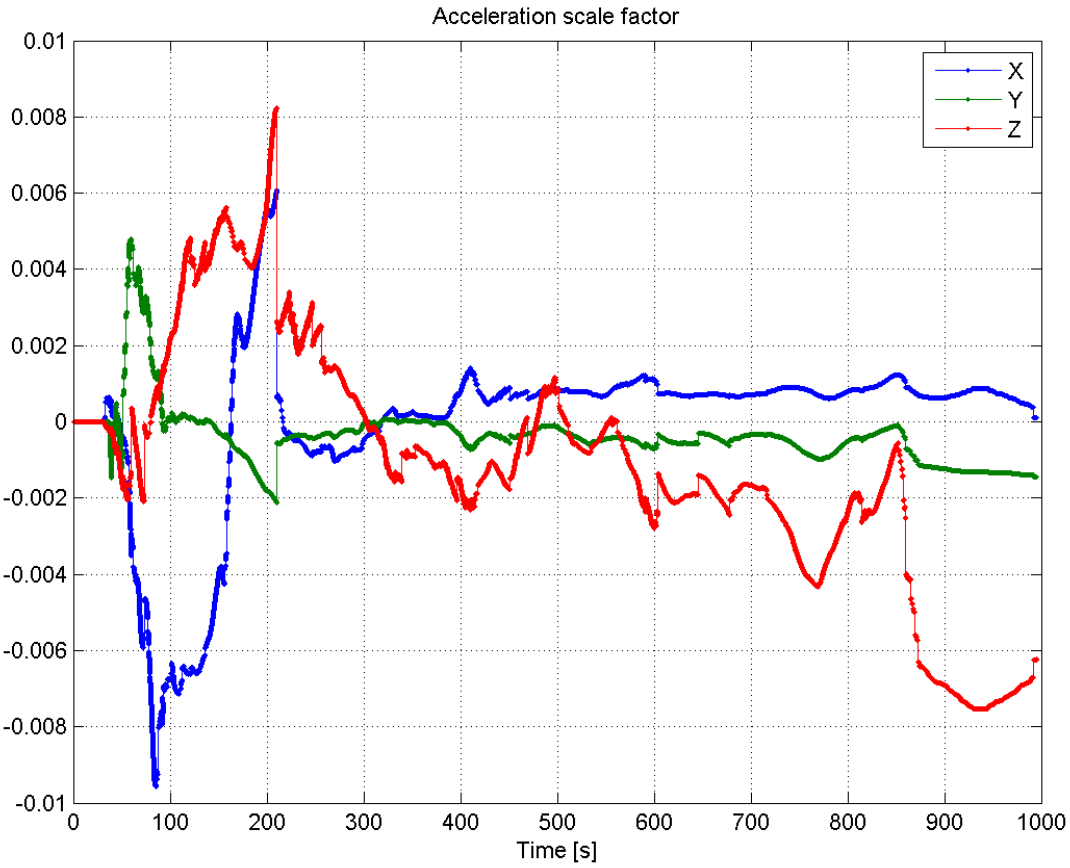



**Figure 9.7** Acceleration scale factor error estimation

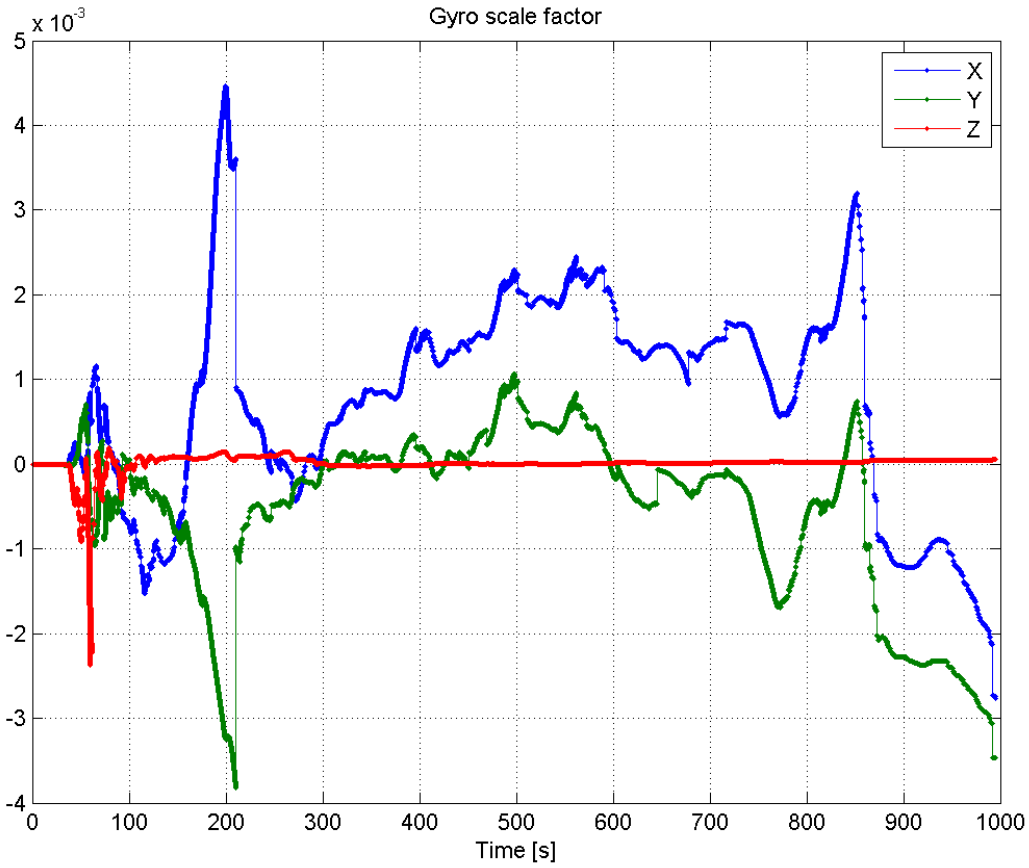



**Figure 9.8** Gyro scale factor error estimation

## 9.3 Software Modules Developed (digital version)

### 9.3.1 AIMS PRO Software

### 9.3.2 Deep Integration Software